Technische Universität Dresden
Fachrichtung Mathematik

Institut für Geometrie

A Nonsmooth Nonconvex Descent Algorithm

Dissertation
zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)

vorgelegt von

Name: Mankau Vorname: Jan, Peter

geboren am: 13.10.1984 in: Aachen

Tag der Einreichung: 05.10.2016

Betreuer: Prof. Friedemann Schuricht
(Technische Universität Dresden)

Contents

Introduction

In many applications nonsmooth nonconvex energy functions, which are Lipschitz continuous, appear quite naturally . Contact mechanics is a classic example. Often, the linear elasticity does not describe the mechanics sufficiently exact and one chooses other models than Hook's law. For example the Mooney-Rivlin and the Neo-Hook's model often describe torsion effects better than Hook's law does, cf. [45]. But for these models the strain energy is not convex. In many cases the contact is treated by penalizing penetration of different bodies. These penalty energies are often chosen nonsmooth. But at latest when the friction between the bodies can not be neglected, the resulting energy functions become nonsmooth. Typically, a nonsmooth Tresca- or Coulomb-friction term is added further to the strain energy term, cf. [37].
A second example is the 1-Laplace operator and its eigenfunctions. The 1-Laplace operator has been studied during the last decades. Let $\Omega \subset \mathbb{R}^n$ be an open and bounded set with Lipschitz boundary and $BV(\Omega) \subseteq L^1(\Omega)$ be the space of functions of bounded variation. A minimizer of

$$F_1(u) := \int_\Omega d|Du| + \int_{\partial\Omega} |u| d\mathcal{H}^{n-1} \to \min! \,, \quad G_1(u) := \int_\Omega |u|(x)dx = 1 \tag{0.1}$$

for $u \in BV(\Omega)$ is called first eigenfunction of the 1-Laplace operator. Note that F_1 and G_1 are nonsmooth. In [33] it was shown by Fridman and Kawohl that the first eigenfunctions of the p-Laplace operator for $p > 1$ converge to some first eigenfunction of the 1-Laplace operator as $p \to 1$ and this eigenfunction is a characteristic function up to scaling. The first eigenfunction of the p-Laplace operator is a minimizer of

$$F_p(u) := \int_\Omega |\nabla u|^p(x)dx \to \min \,, \quad G_p(u) := \int_\Omega |u|^p(x)dx = 1$$

for $u \in W_0^{1,p}(\Omega)$, cf. [33]. Note that F_p and G_p are smooth on $W_0^{1,p}(\Omega)$ for $p > 1$. The 1-Laplace operator is more sophisticated than the p-Laplace operator for more reasons. Eigenfunctions of the 1-Laplace operator are typically not in $W_0^{1,1}(\Omega)$ and often they are not unique. This makes it even more challenging to compute first eigenfunctions of the 1-Laplace operator. Therefore, up to our knowledge, nobody has computed first eigenfunctions of the 1-Laplace operator with Finite Element Methods (FEM). So far Eigenfunctions of the p-Laplace operator have been computed with FEMs only for the the case $p \geq 1.1$, cf. [29].

In the literature one mainly finds four approaches to compute a minimizer of a nonsmooth nonconvex function E.

1. The most common approach is to approximate the nonsmooth function E by a smooth function $\tilde{E}$. This ansatz has its limits as e.g. the p-Laplace operator shows. In the limit case completely new phenomena often occur, which aren't described properly by the limit process. Moreover, classical optimization algorithms often show ill convergence results if $\tilde{E}$ is close to E. Due to the high curvatures of the graph of $\tilde{E}$, the iteration points of many descent algorithms oscillate strongly. We point out that if $\tilde{E}$ is close to E, many algorithms can not distinguish between $\tilde{E}$ and E, since they compute both functions only on a discrete set. In this case the question arises whether it is really the right ansatz to smooth the function E, since the algorithm doesn't distinguish between $\tilde{E}$ and E. One can go even one step further. We observe that up to now nobody has treated the first eigenfunction of the p-Laplace operator numerically for $1 < p < 1.1$ even though the functions are smooth. Maybe it is more successful to treat F_p and G_p as nonsmooth functions and to apply an algorithm for nonsmooth functions. This ansatz could be referred to as "antismoothing", as suggested by Schuricht.

2. It is also a common approach to apply classical smooth algorithms to nonsmooth functions. E.g. Lewis and Overton apply the BFGS algorithm favorable to nonsmooth nonconvex functions, cf. [39]. But as they mention, it is an open problem, whether every accumulation point of BFGS is a critical point in the sense of Clarke. In [39] BFGS was always applied to nonsmooth nonconvex functions $f : \mathbb{R}^n \to \mathbb{R}$ with $n \leq 10$, except for one application, where $n = 2,000$. But in typical FEM applications $n >> 2,000$.

3. In many applications the energy function is nonsmooth in such a way, that one can not exploit any structure at all. In those cases one can use e.g. the Nelder Mead simplex algorithm, cf. [1], and stochastic algorithms like genetic algorithms or LSRS algorithms, cf. [25]. But of course, such algorithms ignore a lot of information in the case that the function is Lipschitz continuous. Information which could be used to improve the speed and robustness of the algorithm.

4. The fourth approach is to design an algorithm for a specific type of nonsmooth nonconvex functions. E.g. in [52] Schramm generalized the bundle-trust region method, which is designed for convex functions, to a special class of Lipschitz continuous, nonsmooth nonconvex functions. Up to our knowledge this algorithm hasn't been tested yet. We also mention here the gradient sampling algorithm which is a robust, stochastic algorithm designed to minimize a locally Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$, which is continuously differentiable on an open and dense set $D \subseteq \mathbb{R}^n$, where n is small, cf. [4]. We also recall that there exists an elaborate theory of algorithms for convex functions, which are robust and fast, cf. [1].

In this work we will give an algorithm such that for every locally Lipschitz continuous function $f : X \to \mathbb{R}$ and every sequence $(x_k)_{k\in\mathbb{N}}$ in X produced by this algorithm it holds that every accumulation point of $(x_k)_{k\in\mathbb{N}}$ is a critical point in the sense of Clarke. Here X is a reflexive Banach space, such that X and its dual space X' are strictly convex and Clarkson's inequalities hold. (E.g. $X := W_0^{1,p}(\Omega)$ and every closed subspace equipped with the Sobolev norm satisfy these assumptions for $p > 1$.) Thus it is left to the user whether he applies the first-discretize-then-optimize or the first-optimize-then-discretize approach. This algorithm is designed primarily to solve variational problems or their high dimensional discretizations, but can be applied to a variety of locally Lipschitz functions.
In elastic contact mechanics the strain energy is often smooth and nonconvex on a suitable domain, while the contact and the friction energy are nonsmooth and have a support on a subspace which has a substantially smaller dimension than the strain energy, since all points in the interior of the bodies have only effect on the strain energy. For such elastic contact problems we suggest a specialization of our algorithm, which treats the smooth part with Newton like methods. In the case that the gradient of the entire energy functions is semismooth close to the minimizer, we can even prove superlinear convergence of this specialization of our algorithm.
We test the algorithm and its specialization with a couple of benchmark problems. Moreover, we apply the algorithm to (0.1) restricted to finitely dimensional subspaces of piecewise affine, continuous functions.

The algorithm developed here uses ideas of the bundle trust region method by Schramm, cf. [52], and a new generalization of the concept of gradients on a set. The gradient on a sets was introduced by Goldstein for functions $f : \mathbb{R}^n \to \mathbb{R}$, where $\mathbb{R}^n$ is equipped with the Euclidean norm, cf. [24]. Gradients on sets are also fundamental for the gradient sampling algorithm, by Burke, Lewis, Kiviel and Overton, cf. [4, 36]. The basic idea behind this gradients on sets is that we want to find a stable descent direction, which is a descent direction on an entire neighborhood of an iteration point. This way we avoid oscillations of the gradients and very small descent steps (in the smooth and in the nonsmooth case). It turns out, that the norm smallest element of the gradient on a set provides a stable descent direction. For the algorithm which we present here, these gradients on sets are a substitute for the $\varepsilon-$subdifferential used in convex optimization. With this concept we can generalize the Armijo step size similar to [52]. A descent step always satisfies the Armijo step size strategy in this work.
The algorithm we present here is the first algorithm which can treat locally Lipschitz continuous functions in this generality, up to our knowledge. In particular that large finitely dimensional Banach spaces haven't been studied for nonsmooth nonconvex functions so far. We will show

that the algorithm is very robust and often faster than common algorithms. Furthermore, we will see that with this algorithm it is the first time possible to compute reliably the first eigenfunctions of the 1-Laplace operator up to disretization errors.
Motivated by these good results we see a lot of potential to apply this algorithm to contact mechanics with friction in the future. Although we have implemented this algorithm for simple contact mechanic problems and gained some promising results, we will not discuss contact problems in full generality in this thesis in order to keep this thesis concise and leave it to later work.

This thesis is structured as follows: In Chapter 1 we first recall the generalized gradient by Clarke. Then we motivate our ansatz to use gradients on sets with tacitly known results from nonsmooth analysis. After that we give our generalization of the gradient on a set for locally Lipschitz continuous functions $f : X \to \mathbb{R}$, where X is a Banach space, which was only defined for locally Lipschitz continuous functions $f : \mathbb{R}^n \to \mathbb{R}$ by Goldstein, cf. [24]. Additionally we will develop a completely new theory for these gradients of sets. We will generalize many results of the generalized gradient by Clarke to the gradients on sets and we will define optimal descent directions on sets and prove under mild assumptions existence and uniqueness of optimal descent directions. Furthermore we prove that if $x \in X$ is not a critical point in the sense of Clarke, then, just as in the smooth case, there exists a neighborhood of x and a direction d which is descent direction on the entity of the neighborhood.
For better readability of the proofs we split the algorithm, which we suggest, into two parts, an outer and an inner algorithm. The outer algorithm calls the inner algorithm to compute some element of the gradient on some set. In Chapter 2 we present the outer algorithm of the optimization algorithm. First we present a simplified version of the algorithm, cf. Algorithm 2.26, to make the reader familiar with the basic ideas. Later we generalize this algorithm to Algorithm 2.38 which solves minimization problems faster. In Algorithm 2.38 we allow a change of the norm in every iteration step, which enables the application of ideas from Newton like methods. Note that the Newton method is an optimal descent step method, if we change the norm in every iteration. Further Algorithm 2.38 is more suitable to minimize functions that are the sum of a smooth and a nonsmooth function, as they appear in contact mechanics.
In Chapter 3 we formulate the inner algorithm, which suggests descent directions for the outer algorithm. The basic idea behind this algorithm is to choose successively better inner approximations of the gradient on a neighborhood (which is typically unknown) until the approximation is such good that the resulting descent direction satisfies the sufficient descent condition by Armijo. We point out that the outer and the inner algorithm are new algorithms, which can not be found in the literature.

In Chapter 4 we show how ideas of semismooth Newton methods can be incorporated into Algorithm 2.38. We show that under reasonable assumptions on the energy function, Algorithm 2.38 is superlinearly convergent, just like the semismooth Newton method.
In Chapter 5 we compare Algorithm 2.38 with a variety of algorithms. We mainly compare with the BFGS algorithm, the bundle trust region method and gradient sampling method. Thus we compare with an algorithm for smooth functions, one for convex functions and a stochastic algorithm designed for nonsmooth nonconvex functions, which are defined on low dimensional Hilbert spaces.
Finally in Chapter 6 we study the 1-Laplace operator for subsets $\Omega \subseteq \mathbb{R}^n$, where either $n = 2$ or $n = 3$. First, we give a rough introduction to the 1-Laplace operator. Then, we show that it is possible to approximate the minimization problem with a finite element approach in spite of the minimizers being an element of $BV(\Omega) \setminus W_0^{1,1}(\Omega)$. We discuss the discretization error and the error resulting from quadrature formulae. We formulate the optimization algorithm in its entity, i.e. Algorithm 2.38 with the inner algorithm. We will make a convergence study of the entire algorithm for the most simple case that the domain Ω is a square, because in this case we know the analytic solution of (0.1). Furthermore, we apply the algorithm to different domains. First we study the case $\Omega \subset \mathbb{R}^2$, because in this case the solution is often known analytically. After that we also apply the algorithm in the more sophisticated situation $\Omega \subset \mathbb{R}^3$, although here the solution is not known analytically. But we can "confirm numerically" a hypothesis about how the solutions should look like in this case. Further we give some discretization error estimates and we also study roughly the eigenfunctions of the p-Laplace operator in the case $p > 1$. For $p > 1$ we additionally compare our algorithm with an algorithm by J. Horák , cf. [29].

In summery we can say that we create an algorithm, which is capable to find a minimizer of a nonsmooth nonconvex function, which is locally Lipschitz continuously and defined on a reflexive Banach space on which Clarkson's inequalities hold. This algorithm can compute numerically the first eigenfunction of the 1-Laplace operator. No other algorithm managed to compute them till now. Therefore we think that this algorithm should be tested further with energy functions from contact mechanics.

Acknowledgments

First of all I would like to thank my supervisor Professor Schuricht for the opportunity to work on this thesis. Without his encouraging guidance, inexhaustible patience and his persistent help this thesis would have been not possible. The many hours at the phone are very much appreciated and not taken for granted. They helped tremendously to improve the quality of results, their presentation and readability. These discussions and advices will be a precious guidance for upcoming works.

Moreover I would like to thank my colleagues. Foremost I thank Nadine Albrecht and Moritz Schönherr for the inspiring discussions which influenced this work deeply. Further I am grateful to Zoja Milbers and Samuel Littig for our debates on the 1-Laplace operator.
Additionally I am thankful to the members of the faculty of mathematics for useful advice, fascinating discussions and providing a productive atmosphere to study.

1 Optimal Descent Directions

In this chapter X always denotes a Banach space and $f : X \to \mathbb{R}$ a locally Lipschitz continuous function, i.e. for every point $x \in X$ there exists a neighborhood $U(x)$ such that f restricted to $U(x)$ is Lipschitz continuous. We will define and determine optimal descent directions. For this we need a concept of a gradient for nonsmooth functions. A possible concept for this is the generalized gradient by Clarke, which we will introduce now.

1.1 Clarkes Generalized Gradient

Of course, there are many ways to generalize the concept of a gradient to locally Lipschitz continuous functions, which lead to different concepts of a generalized gradient. We follow here the approach by Frank H. Clarke. Following his papers, one can of course see that the definition has changed until its optimal form has been founded, which was written down in Clarke's famous book [13]. Here we give only the main definitions and results of [13] without proofs, which can be found in [13].

Definition 1.1 Let x and v be any vectors in X. The **generalized directional derivative of f in the direction** v is defined as

$$f^0(x; v) := \limsup_{y \to x,\ t \downarrow 0} \frac{f(y + tv) - f(y)}{t}, \tag{1.2}$$

where $y \in X$ and t is a positive scalar.

We observe this expression always exists and it reflects the derivatives of the gradients approaching x. If f is smooth this is of course the classical directional derivative. This derivative is useful for our further analysis because of the following proposition.

Proposition 1.3 (upper semicontinuity)
Let $f : U(x) \subset X \to \mathbb{R}$ be Lipschitz continuous with Lipschitz constant K, where $U(x)$ is a neighborhood of $x \in X$. Then

1. *The function $v \to f^0(x; v)$ is finite, positively homogeneous, and subadditive on x, and satisfies*
$$\left|f^0(x; v)\right| \leq K \|v\|.$$
2. *$f^0(x; v)$ is upper semicontinuous as a function of (x, v) and, as a function of v alone, is Lipschitz of rank K on X.*
3. *$f^0(x; v) = (-f)^0(x, -v)$.*

PROOF. This has been proved in [13, Proposition 2.1.1]. ♢

With this we can define the generalized gradient.

Definition 1.4 The **generalized gradient of f at** x is the subset of the dual space X' of X given by

$$\partial f(x) := \left\{ f' \in X' \,\middle|\, f^0(x;v) \geq \left\langle f' \,\middle|\, v \right\rangle \text{ for all } v \in X \right\}. \tag{1.5}$$

The generalized gradient describes the generalized directional derivative.

Proposition 1.6 (characterization of the generalized gradient)
Let f be Lipschitz of rank K near x. Then

1. *$\partial f(x)$ is a nonempty, convex, weak*-compact subset of X' and*

$$\left\| f' \right\|_{X'} \leq K$$

   for every $f' \in \partial f(x)$.

2. *For every $v \in X$, one has*

$$f^0(x;v) = \max \left\{ \left\langle f' \,\middle|\, v \right\rangle \;\middle|\; f' \in \partial f(x) \right\}.$$

3. *$f' \in \partial f(x)$ iff $f^0(x;v) \geq \left\langle f' \,\middle|\, v \right\rangle$ for all $v \in X$.*

PROOF. The proof is in [13, Proposition 2.1.2 and Proposition 2.1.5]. ♢

Of course we can not expect ∂f to be continuous, but we gain the next best. If we understand ∂f as the relation $\left\{ (x, f') \in X \times X' \,\middle|\, f' \in \partial f(x) \right\}$, then ∂f is closed in $X \times X'$, if X' is endowed with the weak*-topology. This is formulated in the following proposition.

Proposition 1.7 (upper semicontinuity)
Let $f : X \to \mathbb{R}$ be locally Lipschitz continuous.

1. *Let $(x_i)_{i\in\mathbb{N}}$ and $(f_i')_{i\in\mathbb{N}}$ be sequences in X and X' such that $f_i' \in \partial f(x_i)$. Suppose $(x_i)_{i\in\mathbb{N}}$ converges to x and that f' is a cluster point of f_i' in the weak*-topology. Then one has $f' \in \partial f(x)$.*

2. $$\partial f(x) = \bigcap_{\delta>0} \bigcup_{y \in B(x\,,\,\delta)} \partial f(y).$$

3. *If X is finite-dimensional, then ∂f is upper semicontinuous at x.*

PROOF. This result has been proved in [13, Proposition 2.1.5]. ♢

Another way to see Proposition 1.7 is to say that ∂f is upper semicontinuous in the Hausdorff metric.
A big advantage of Clarkes generalized gradient is that it is really a generalization of many concepts of gradients.

Proposition 1.8 (differentiable functions)
Let $f : X \to \mathbb{R}$ be continuous.

1. *Let f be Lipschitz near x and admit a Gâteaux derivative $Df(x)$. Then $Df(x) \in \partial f(x)$.*
2. *If f is strictly differentiable at x, then f is Lipschitz near x and $\partial f(x)$ consists only of the strict derivative.*
   Conversely, if f is Lipschitz near x and $\partial f(x)$ reduces to a singleton $\{f'\}$, then f is strictly differentiable at x and the strict derivative at x is f'.
3. *When f is convex and Lipschitz continuous at the neighborhood $U(x)$ of x, then $\partial f(x)$ coincides with the subdifferential at x in the sense of convex analysis, and $f^0(x; v)$ coincides with the directional derivative $f'(x; v)$ for each v.*

PROOF. This proposition is just a merge of results in [13, Proposition 2.2.2, Proposition 2.2.4 and Proposition 2.2.7]. ♢

Next we give a necessary condition for an extremum and some calculation rules.

Proposition 1.9 (extrema)
If $f : X \to \mathbb{R}$ is Lipschitz continuous and attains a local minimum or maximum at x, then

$$0 \in \partial f(x).$$

PROOF. The proof can be found in [13, Proposition 2.3.2]. ♢

Proposition 1.10 (sum of functions)
Let $f : X \to \mathbb{R}$ and $f_i : X \to \mathbb{R}$ be locally Lipschitz continuous.

1. *For any $s \in \mathbb{R}$ we have $\partial(sf)(x) = s\,\partial f(x)$.*
2. $$\partial\left(\sum_{i=1}^{n} f_i\right)(x) \subseteq \sum_{i=1}^{n} \partial f_i(x)$$
   and equality holds if all but at most one of the functions f_i are strictly differentiable at x.

PROOF. Cf. [13, Proposition 2.3.1, Proposition 2.3.3 and the related Corollary 1]. ♢

A result we will use later is a generalization of the classical Mean-Value Theorem, the so called Theorem of Lebourg.

Proposition 1.11 (Lebourg's Theorem)
Let x and y be vectors in X, and suppose that f is Lipschitz on an open set containing the line segment $[x, y]$. Then there exists a point u in (x, y) such that

$$f(y) - f(x) \in \langle \partial f(u) \,|\, y - x \rangle . \tag{1.12}$$

PROOF. The proof of Lebourg's Theorem can be found in [13, Theorem 2.3.7]. ♢

For further calculation rules, we need the concept of regularity.

Definition 1.13 $f : X \to \mathbb{R}$ is said to be **regular at** x provided

1. For all $v \in X$ the usual one-sided directional derivative $f'(x; v)$ exists.
2. For all $v \in X$ it is $f'(x, v) = f^0(x; v)$.

We give some examples of regular functions.

Proposition 1.14 (examples of regular functions)
Let f be Lipschitz near x.

1. *If f is strictly differentiable at x, then f is regular at x.*
2. *If f is convex, then f is regular at x.*
3. *A finitely linear combination (by nonnegative scalars) of functions, which are regular at x, is regular at x.*
4. *If f admits a Gâteaux derivative $Df(x)$ and is regular at x, then*

$$\partial f(x) = \{Df(x)\}$$

PROOF. Also this is a well known result, cf. [13, Proposition 2.3.6]. ♢

With this we can formulate the product and the quotient rule, which we will need to treat the p-Laplacian.

Proposition 1.15 (product and quotient rule)
Let f_1, f_2 be Lipschitz near x.

1. *Then $f = f_1 f_2$ is Lipschitz near x, and one has*

$$\partial f(x) \subseteq f_2(x)\partial f_1(x) + f_1(x)\partial f_2(x).$$

If in addition $f_1(x) \geq 0, f_2(x) \geq 0$ and if f_1, f_2 are both regular at x, then equality holds and f is regular.

2. *Suppose $f_2(x) \neq 0$. Then $f = f_1/f_2$ is Lipschitz near x, and one has*

$$\partial f(x) \subseteq \frac{f_2(x)\partial f_1(x) - f_1(x)\partial f_2(x)}{(f_2(x))^2}.$$

If in addition $f_1(x) \geq 0, f_2(x) > 0$ and if f_1 and $-f_2$ are both regular at x, then equality holds and f is regular at x.

PROOF. The proof can be found in [13, Proposition 2.3.13 and Proposition 2.3.14]. ♢

We would like to create penalty functions. For this purpose, we give now a chain rule.

Proposition 1.16 (chain rule)
Suppose $h : X \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}$ are Lipschitz continuous and g is regular at $h(x)$, every component function h_i of h is regular at x and every element of $\partial g(h(x))$ has only nonnegative components. Then one has for $f = g \circ h$

$$\partial f(x) = \overline{conv}\left\{\sum_{i=1}^{n} \alpha_i f_i' \;\middle|\; f_i' \in \partial h_i(x),\, \alpha_i \in \partial g(h(x))\right\} \tag{1.17}$$

and f is regular at x.

PROOF. The chain rule has been proved in [13, Proposition 2.3.9]. ♢

Next we give the generalized gradient of the pointwise maximum of functions.

Proposition 1.18 (pointwise maxima)
Suppose now $f_1, \ldots, f_n$ are locally Lipschitz continuous functions and

$$f(x) = \max_{i=1,\ldots,n} f_i(x) \quad \textit{for}\, x \in X\ .$$

If we denote by $I(x) = \{i \mid f(x) = f_i(x)\}$ the set of active indeces, we have

$$\partial f(x) \subseteq conv\{\partial f_i(x) \mid i \in I(x)\}\ , \tag{1.19}$$

and if the f_i are regular at x, the equality holds and f is regular at x.

PROOF. Cf. [13, Proposition 2.3.12]. ♢

At last we give a result which helps to understand the generalized gradient in the finite dimensional case. We recall first Rademacher's famous theorem, which states that a Lipschitz continuous function $f : \mathbb{R}^n \to \mathbb{R}$ is almost everywhere differentiable.

Proposition 1.20 (finite dimensional X)
Let $f : \mathbb{R}^n \to \mathbb{R}$ be locally Lipschitz continuous. Suppose $S \subset \mathbb{R}^n$ is a set of Lebesgue measure 0 containing all points of f in which f is not differentiable. Then

$$\partial f(x) = conv\left\{ \lim_{i\to\infty} \nabla f(x_i) \;\middle|\; (x_i)_{i\in\mathbb{N}} \in (\mathbb{R}^n \setminus S)^{\mathbb{N}} \text{ with } x_i \to x \right\} . \tag{1.21}$$

PROOF. We refer to [13, Theorem 2.5.1]. ♢

1.2 Optimal Descent Direction

Next we give a definition of an optimal descent direction. In basic analysis and numeric courses the first idea is to take minus the gradient of the smooth function. It is well known that this gives a descent direction which is also in some sense optimal. This so called Cauchy method or the method of Steepest Descent was suggested by Cauchy and has been studied intensively; we name here e.g. the works of A. Cauchy and A. Goldstein, [6, 22, 23]. From the analytical point of view this is a very fertile idea, which leads to very strong and powerful results to find (locally) minimal points even in the nonsmooth case. For further results we recommend e.g. the book of R. Chill [8]. But using just the gradient at a point to construct a descent direction for a numerical method leads to very slow algorithms, which might not even give a converging sequence and if the sequence converges, it converges not to a critical point, cf. W. Alt [1, 2]. A problem arises from the fact that a line-search only approximates a minimizer and that in the case of a nonsmooth energy function (or a function with high curvature) the gradient (the optimal descent direction) varies strongly. This leads to strong oscillation. Take for example the function $f : \mathbb{R}^2 \to \mathbb{R}$ with $f(x_1, x_2) = |x_1| + \varepsilon|x_2|$ with $0 < \varepsilon << 1$ and as initial point a point $(x_1^0, x_2^0) \approx (0, 10)$ but $(x_1^0, x_2^0) \neq (0, 10)$. Then the gradient at a point (x_1, x_2) close to (x_1^0, x_2^0) with $x_1 \neq 0$ is either $(1, \varepsilon)$ or $(-1, \varepsilon)$. So the numerical implementation of the Steepest Descent method oscillates around the axis $\{0\} \times \mathbb{R}$ as we can see in Figure 1. The gradient flow, as solution of a gradient system, would not show this behavior.

We would like to mention that this function is not just of theoretical interest. To the contrary, it is even to some extent the common case. Observe that by Rademacher's Theorem, a Lipschitz continuous function is almost everywhere differentiable, i.e. we can locally linearize the function

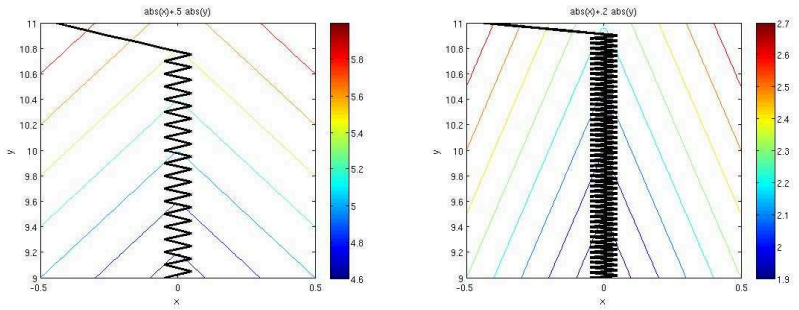

Figure 1: The level sets of the function $f(x,y) = |x| + \varepsilon|y|$, with $\varepsilon = 0.5$ and $\varepsilon = 0.2$, and the iterations of the Steepest Descent method with $x_{k+1} = x_k - 0.1f'(x_k)$. One can show that decreasing the step size in every step even increases the number of necessary gradients to reach the same distance of the minimizer 0 and the approximation point x_k. Therefore we keep the step size constant $0.1 \cdot \|f'(x,y)\|$

almost everywhere. This means around the nonsmooth points, we can approximate the function by piecewise linear functions very good.

To solve the above problems, various concepts have been developed to improve the idea of the Cauchy Method. In the case of convex functions, successful methods are the Bundle-Methods using the subdifferential and the ε-subdifferential as developed by H. Schramm, cf. [2, 52, 58]. Basically the idea in these algorithms is to use the gradients of former iteration points to gain an inner approximation of the ε-subdifferential. The $\varepsilon-$ subdifferential is then used to define a descent direction of a neighborhood of the current iteration point. This algorithm benefits to a large amount of the fact that one can calculate the minimal point of a function, which is the maximum of finitely many linear functions, analytically. So the idea is to approximate the function f locally by a model function, which is the maximum of the functions

$$f_k(x) := f(x_k) + \left\langle \nabla f(x_k) \,\middle|\, x - x_k \right\rangle \ ,$$

where x_k is the k-th iteration point and $\nabla f(x_k)$ is a subgradient in x_k. An advantage of this approach is that the $\varepsilon-$ subdifferential of f in x_k can be approximated by the subset, which is given by the $\varepsilon-$subdifferential of $\max_{i \le k} f_i$ in x_k, cf. [2, Beispiel 5.15]. This is combined with ideas of Trust-Region Methods as collected in the work of A. Conn, N. Gould and P. Toint, [15].

The ideas of Trust-Region Methods have been also used to create other algorithms which approximate the function on a trusted region by a nonsmooth model function of which one can compute the Cauchy point. In [17, 41, 48] one could prove convergence with the concepts of

the collection [15] under special assumptions on the model function. But these assumptions are quite harsh and too harsh for our situation, since one needs that the generalized gradients of the energy and the model function at a point coincide in a kind of smooth way. Up to our knowledge this approach has been only used to minimize functions which are the maximum or minimum of finitely many smooth functions.

We also want to mention the semismooth Newton method here, cf. M. Ulbrich [56, 57]. This wellknown concept gives very good results. But one needs that we start sufficiently close to the solution, the first derivative is semismooth around the solution, bounded and the inverse is bounded too. These are all assumptions, which are too strong for our situation. But we will see that the semismooth Newton method might be seen as a special case of our algorithm if the assumption of the semismooth Newton method are satisfied. But we achieve global convergence, cf. chapter 4.

Another idea is to define the "gradient of a given neighborhood" of a iteration point abstractly. This gives us, similar to the common steepest descent direction, a "steepest descent direction at the neighborhood". Goldstein has used this idea to create a converging theoretical algorithm in [24] for functions $f : \mathbb{R}^n \to \mathbb{R}$. This ansatz was used by J. Burke, A. Lewis, M. Overton and K. Kiwiel in [4, 36] to develop an implementable algorithm, the Gradient-Sampling algorithm, which tries to compute an inner approximation of the "gradient of a given neighborhood" by randomly sampling gradients in this neighborhood. We will also use the "gradient of the neighborhood" and the related "steepest descent direction at the neighborhood". For this we will now present a theory for these concepts. Our aim is to approximate the "gradient of the neighborhood" with suitable subsets, which are not constructed randomly. We will use the ideas from the work of H. Schramm, [52]. However, we do not use the ε-subdifferential, so we will not present the theory here, although the concepts can be easily transfered.

First we define descent directions and observe some simple results.

Definition 1.22 We call $d \in X$ a **descent direction of f at the point** $x \in X$, if there exists some $t_0 > 0$ such that for all $0 < t \leq t_0$

$$f(x) > f(x + td).$$

d is called **descent direction of f on** $M \subseteq X$, if d is a descent direction of f at every point $x \in M$.

Next we give a sufficient condition for descent directions. First we concentrate on the descent directions at one point to keep the formulations simple, because first we just motivate our approach. Later the results for the descent direction of a set follow directly.

Proposition 1.23 (sufficient condition)
Let $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Suppose $x, d \in X$ satisfy

$$\begin{aligned} & f^0(x; -d) < 0 \\ \Leftrightarrow & \left\langle \partial f(x) \,\middle|\, d \right\rangle \subseteq \mathbb{R}_{>0} \,, \end{aligned}$$

then $-d$ is a descent direction of f at the point x.

Proof. This follows directly from

$$\begin{aligned} \limsup_{t \searrow 0} \frac{f(x - td) - f(x)}{t} &\leq f^0(x; -d) \\ &= \max \left\langle \partial f(x) \,\middle|\, -d \right\rangle = -\min \left\langle \partial f(x) \,\middle|\, d \right\rangle < 0 \,, \end{aligned}$$

where the last inequality is strict, because of the weak* compactness of $\partial f(x)$, which implies that the minimum is attained. ♢

Remark 1.24 In the smooth case this gives that if $\nabla f(x) \neq 0$, thus $\partial f(x) = \{\nabla f(x)\}$, we even gain an entire open half-plane of descent directions. This is not necessarily true in the nonsmooth case. Consider for example again the function $f(x, y) := |x| + |y|$ and $(x, y) := (1, 0)$. Then the cone $\left\{ s(-1, \lambda) \,\middle|\, s > 0,\ \lambda \in (-1, 1) \right\}$ is the set of all descent directions, which is no a half-plane.
We also want to mention here that the set of descent directions is by definition independent of the norm.

This sufficient condition gives us immediately the existence of a descent direction in the case that $0 \notin \partial f(x)$.

Proposition 1.25 (existence of descent direction)
Let $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Suppose for $x \in X$ we have $0 \notin \partial f(x) \subset X'$, then there exist some $\tilde{d} \in X$ with

$$f^0(x; -\tilde{d}) < 0 \quad \text{and} \quad \left\langle \partial f(x) \,\middle|\, \tilde{d} \right\rangle \subseteq \mathbb{R}_{>0} \,,$$

this means $-\tilde{d}$ is a descent direction of f at the point x.

PROOF. We recall that by definition $\tilde{f}' \in \partial f(x)$ iff

$$f^0(x;d) \geq \left\langle \tilde{f}' \,\middle|\, d \right\rangle \quad \text{for all } d \in X \ .$$

By Proposition 1.6 $0 \notin \partial f(x)$ implies that for some fixed $d \in X$ it holds

$$\max_{f' \in \partial f(x)} \left\langle f' \,\middle|\, d \right\rangle = f^0(x;v) < \left\langle 0 \,\middle|\, d \right\rangle = 0 \ .$$

Now we simply choose $\tilde{d} := -d$. ♢

Next we give a motivating example of how to compute such a descent direction in the simple case that X is a Hilbert space. In the smooth case, this would be the steepest descent direction.

Proposition 1.26 (sufficient condition in Hilbert spaces)
Let X be a real Hilbert space and $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Suppose there exist some $x \in X$ and some $f_0' \in \partial f(x)$ with

$$\min_{f' \in \partial f(x)} \left\| f' \right\| = \left\| f_0' \right\| \neq 0 \ . \tag{1.27}$$

Then for $d = f_0'$ we have

$$f^0(x;-d) < 0 \quad \textit{and} \quad \left\langle \partial f(x) \,\middle|\, d \right\rangle \subseteq \mathbb{R}_{>0} \ .$$

PROOF. We recall first that for every convex $K \subset X$ and every $x \in X$ we have

$$\begin{aligned} & \|x\| = \min \left\{ \|y\| \ \middle|\ y \in K \right\} \\ \Leftrightarrow\ & \forall_{y \in K} \left\langle x - y \,\middle|\, x \right\rangle \leq 0 \\ \Leftrightarrow\ & \forall_{y \in K} \left\langle x \,\middle|\, x \right\rangle \leq \left\langle x \,\middle|\, y \right\rangle \ . \end{aligned}$$

This gives the claim, with $K := \partial f(x)$ and $x = f_0'$.
For completeness we also prove the equivalences: We assume first that $\|x\| = \min \left\{ \|y\| \ \middle|\ y \in K \right\}$, then for every $y \in K$ and $\lambda \in]0,1[$ we have:

$$\begin{aligned} \lambda^2 \|y - x\|^2 + 2\lambda \left\langle x \,\middle|\, y - x \right\rangle = \|x + \lambda(y-x)\|^2 - \|x\|^2 \geq 0 \\ \Leftrightarrow \left\langle x \,\middle|\, y - x \right\rangle \geq -\frac{\lambda}{2} \|y - x\|^2 . \end{aligned}$$

Taking $\lambda \to 0$ gives $\langle x \,|\, y - x \rangle \geq 0$. The other implication follows directly from

$$(\|x\| - \|y\|)\,\|x\| \geq \|x\|^2 - \langle y \,|\, x \rangle = \langle x - y \,|\, x \rangle \leq 0.$$

$\diamondsuit$

We will now deal with the task to generalize Proposition 1.26 to Banach spaces. Since $\partial f(x)$ is convex and weak* compact, we can always find a solution $f_0' \in \partial f(x)$ of the generalization

$$\min_{f' \in \partial f(x)} \|f'\|_{X'} = \|f_0'\|_{X'} \neq 0\,. \tag{MinC}$$

of equation (1.27) to arbitrary Banach spaces. Our choice of d was such that

$$\left\langle f_0' \,\middle|\, \frac{d}{\|d\|} \right\rangle = \|f_0'\| = \sup\left\{ \langle f_0' \,|\, x \rangle \;\middle|\; x \in X,\ \|x\| = 1 \right\}\,, \tag{DualC}$$

thus d would be a predual of f_0' in the case that X is a Banach space, if d exists.

Definition 1.28 Let X be a Banach space and X' the dual space of X. For $x \in X$ and $x' \in X'$ we call x' dual element of x if

$$x'(x) = \|x\| \quad \text{and} \quad \|x'\| = 1$$

and we call x predual element of x' if

$$x'(x) = \|x'\| \quad \text{and} \quad \|x\| = 1\,.$$

If X is reflexiv, we do not distinguish between dual and predual elements and just call it dual element.

Unfortunately we do not get comparable statements for arbitrary Banach spaces. We will even see that not every $-d$, such that d satisfies (DualC), is a descent direction. But if X is reflexive, we find at least one predual d of some $f_0' \in \partial f(x)$ with (DualC) such that $-d$ is a descent direction. If we ensure further that f_0' and d are uniquely determined by (MinC) and (DualC) expect for the norm of d, we can even calculate these vectors up to a scalar. This descent direction turns out to be optimal in the sense that the generalized directional derivative is minimal in this direction. Now we give the definition to generalize the concept of steepest descent direction at a point to Lipschitz continuous functions and prove the existence of such directions. We recall the notation of the unit sphere $S_X(0,1) := \left\{ x \in X \;\middle|\; \|x\| = 1 \right\}$, where $\|\cdot\|$ is a norm on the Banach space X.

Definition 1.29 (optimal descent direction)
Let X be a Banach space and $f : X \to \mathbb{R}$ be Lipschitz continuous near $x \in X$ and $0 \notin \partial f(x)$. We call $d_0 \in S_X(0,1)$ a **generalized steepest** (or **optimal**) **descent direction of** f **in** $x \in X$ **related to the norm** $\|\cdot\|$, if d_0 minimizes the generalized directional derivative, i.e.

$$f^0(x; d_0) = \inf_{\|d\|=1} f^0(x; d) \ .$$

Remark 1.30 As in the smooth case the optimal descent direction depends on the norm. In this chapter we do not study the question: "What is an optimal or good norm". Here we assume that the norm is given. But of course, as soon as it comes to numerical implementation, the norm becomes important. Then the right choice depends on several aspects. First: How much computational time is needed to compute one derivative sufficiently exact and how exact can we represent a derivative. But a second important requirement in our algorithms will be that the dual mapping is also easy to compute and that we do not have to compute too many gradients. If the norm is clear from the context, we will omit the term "related to the norm $\|\cdot\|$."

Now the existence of optimal descent directions follows from the Duality Theorem.

Proposition 1.31 (existence of optimal descent directions)
Let X be a Banach space and $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Then for all $x \in X$ hold:

1. $$\inf_{\|d\|\leq 1} f^0(x; d) = - \inf_{f' \in \partial f(x)} \left\|f'\right\| \ . \tag{1.32}$$

2. *For every pair $(f'_0, d_0) \in \partial f(x) \times \overline{B_X(0\,,\,1)}$ such that*

$$\left\|f'_0\right\| = \min_{f' \in \partial f(x)} \left\|f'\right\| \quad \textit{and} \quad f^0(x; d_0) = \inf_{\|d\|\leq 1} f^0(x; d) \tag{1.33}$$

   holds

$$-\left\|f'_0\right\| = \left\langle f'_0 \,\middle|\, d_0 \right\rangle = f^0(x; d_0) = \sup_{f' \in \partial f(x)} \left\langle f' \,\middle|\, d_0 \right\rangle \ . \tag{1.34}$$

3. *If X is reflexive then a pair $(f'_0, d_0) \in \partial f(x) \times \overline{B_X(0\,,\,1)}$ satisfying (1.33) exists.*

Corollary 1.35 *Let X be a Banach space and $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Then for all $x \in X$ hold:*

$$\inf_{\|d\|\leq 1} f^0(x; d) < 0 \Leftrightarrow 0 \notin \partial f(x)$$

and in this case for every pair $(f_0', d_0) \in \partial f(x) \times \overline{B_X(0\,,\,1)}$ *satisfying (1.33) we have* d_0 *is an optimal descent direction of* f *in* x*.*

Proof Propostion 1.31. From Proposition 1.6 it follows

$$\inf_{\|d\|\leq 1} f^0(x;d) = \inf_{\|d\|\leq 1} \max_{f'\in\partial f(x)} \left\langle f' \,\middle|\, d \right\rangle .$$

Aubin's Lopsided Minimax Theorem [3, Theorem 7,Ch- 6, Sec. 2] tells us that we can exchange the min and the max, since $\partial f(x)$ is weak* compact. So we continue

$$\inf_{\|d\|\leq 1} \max_{f'\in\partial f(x)} \left\langle f' \,\middle|\, d \right\rangle = \max_{f'\in\partial f(x)} \inf_{\|d\|\leq 1} \left\langle f' \,\middle|\, d \right\rangle = \max_{f'\in\partial f(x)} -\left\|f'\right\| \ ,$$

which gives us (1.32).
Now we assume there exists a pair $(f_0', d_0) \in \partial f(x) \times \overline{B_X(0\,,\,1)}$ satisfying (1.33). By assumption and (1.32) it holds $-\|f_0'\| = f^0(x;d_0)$ and so the estimates

$$f^0(x;d_0) = \sup_{f'\in\partial f(x)} \left\langle f' \,\middle|\, d_0 \right\rangle \geq \left\langle f_0' \,\middle|\, d_0 \right\rangle \geq \inf_{\|d\|\leq 1} \left\langle f' \,\middle|\, d \right\rangle = -\left\|f_0'\right\| \ ,$$

give the claimed equalities.
In the case X is reflexiv the Duality Theorem, cf. [60, Theorem 49b], gives us the existence of a saddle point of the minmax problem

$$\inf_{\|d\|\leq 1} \max_{f'\in\partial f(x)} \left\langle f' \,\middle|\, d \right\rangle = \max_{f'\in\partial f(x)} \inf_{\|d\|\leq 1} \left\langle f' \,\middle|\, d \right\rangle \ ,$$

since all sets are bounded. ♢

The following example shows that the assumption, X is reflexive, is a necessary assumption for Proposition 1.31 in the sense that we can not drop it without making other assumptions.

Example 1.36 (counter example for nonreflexive Banach space)
We choose X *as the Banach space of all null sequences in* $\mathbb{R}$

$$X = c_0 := \left\{ (x_i)_{i\in\mathbb{N}} \in \mathbb{R}^{\mathbb{N}} \,\middle|\, \lim_{i\to\infty} x_i = 0 \right\}$$

with the supremum norm. Then the dual space is given by

$$X' = l_1(\mathbb{N}) := \left\{ (x_i)_{i\in\mathbb{N}} \in \mathbb{R}^{\mathbb{N}} \,\middle|\, \sum_{i\in\mathbb{N}} |x_i| < \infty \right\}$$

with the usual l_1 norm, cf. D. Werner, [59, Satz II.2.3 (b)]. Further we consider the linear and continuous (hence Lipschitz continuous) function $f = (\frac{1}{2^{i+1}})_{i\in\mathbb{N}} \in X'$ with $f(x) = \sum_{i\in\mathbb{N}} \frac{x_i}{2^{i+1}}$ and $\|f\| = 1$. Then in all points $x \in X$ it is $\partial f(x) = \{f\}$. But there exists no $d_0 \in X$ with $\|d_0\| = 1$ und $\langle f \,|\, d_0\rangle = -\|f\|$. Thus, with this choice of X and f, Proposition 1.31 can not be true. We can not find an optimal descent direction.
However we of course find some descent directions d, which gives $\langle f \,|\, d\rangle$ is arbitrarily close to -1. We just have to consider the sequence of descent directions[1] $(d_n)_{n\in\mathbb{N}} \in X^{\mathbb{N}}$ with $(d_n)_i = -1$ for $i \le n$ and $(d_n)_i = 0$ else. Then $\|d_n\|_\infty = 1$ and $\langle f \,|\, d_n\rangle \to -1$.

So now we know that if X is reflexive there exists at least one optimal descent direction and every optimal descent direction is minus a dual of some $f_0' \in \partial f(x)$ with (MinC). But this is not a general characterization as we will see in the next example! We show that there exist $f : X \to \mathbb{R}$ and $x, d \in X$ and $0 \neq f_0' \in \partial f(x)$ with (MinC), $\|d\| = 1$ and $-\|f_0'\| = \langle f_0' \,|\, d\rangle$, thus $-d$ is a dual of f_0', but d and $-d$ are not descent directions.

Example 1.37 (counter example for a not strictly convex norm)
We choose $X := \mathbb{R}^2$ and for $\alpha, \beta, \gamma, \delta \in \mathbb{R}$ we define $f : \mathbb{R}^2 \to \mathbb{R}$ by

$$(x, y) \to |\alpha x - \beta y| + \gamma x + \delta y.$$

Then we have

$$\partial f((x,y)) = \left\{ \begin{pmatrix} \gamma \\ \delta \end{pmatrix} \right\} + \begin{cases} \left\{ \begin{pmatrix} \alpha \\ -\beta \end{pmatrix} \right\} & \text{for } \alpha x > \beta y \\ \left\{ \begin{pmatrix} -\alpha \\ \beta \end{pmatrix} \right\} & \text{for } \alpha x < \beta y \\ \left\{ t \begin{pmatrix} -\alpha \\ \beta \end{pmatrix} \;\middle|\; t \in [-1,1] \right\} & \text{for } \alpha x = \beta y \end{cases}$$

Now we choose the norm on X.

1. *$\|\cdot\|_1$: If we choose the $1-$norm $\|(x,y)\|_1 := |x| + |y|$ on X, then X' is equipped with the supremum norm $\|\cdot\|_\infty$. If we choose now e.g. $\alpha = 0$, $-\beta = -1$, $\gamma = 1$, $\delta = 0$, thus*

$$f(x, y) = x + |y| \,,$$

[1] $\langle f \,|\, -d_n\rangle > 0$

and $x = 1$, $y = 0$, then for all $f' \in \partial f(x,y) = \{(1,t) \mid t \in [-1,1]\}$ we have

$$\|f'\|_{X'} = 1.$$

Further the choices $d := (0,-1)^T$ and $f_0' := (1,1)^T \in \partial f(x,y)$ give

$$\langle f_0' \mid d \rangle = -\|f_0'\|_{X'} \text{ and } \|d\|_X = 1.$$

But in both directions d and $-d$ the function is strictly increasing.
Here the descent direction of Proposition 1.31 would be $d_0 := (-1,0)$, which satisfies

$$\langle f' \mid d_0 \rangle = -\|f'\|_{X'} \quad \text{for all } f' \in \partial f(x,y)\ .$$

2. *$\|\cdot\|_\infty$: If we choose the supremum norm $\|\cdot\|_\infty$ on X, then X' has the $1-$norm $\|\cdot\|_1$. If we choose now e.g. $\alpha = \beta = \gamma = \delta = \frac{1}{2}$, thus*

$$f(x,y) = \frac{1}{2}(x + y + |x - y|)\ ,$$

and $x = y$, then we have

$$f_0' := (1,0)^T \in \partial f(x,y) = \{(\lambda, 1-\lambda) \mid \lambda \in [0,1]\}$$

and for all $f' \in \partial f(x,y)$

$$\|f'\|_{X'} = 1.$$

Further it holds with the choice $d := (-1,1)^T$ on the one hand $\|d\|_X = 1$ and on the other hand $\langle f_0' \mid d \rangle = -\|f_0'\|_{X'}$. But in both directions d and $-d$ is the function strictly increasing.
Here we did not choose f_0' properly. If we choose $f_0' := \frac{1}{2}(1,1)$ then $d_0 := -(1,1)$ is uniquely determined by $\|d_0\|_X = 1$ and $\langle f_0' \mid d_0 \rangle = -\|f_0'\|_{X'}$ and d_0 is our optimal descent direction.

We recall that by Proposition 1.31 there exists an optimal descent direction which is also minus a dual of the solution f_0' of (MinC) in the case that X is reflexive. Therefore we assume in the following that X is reflexive. Since we do not have an efficient method to compute $f^0(x;v)$, we do not know how to compute the optimal descent direction directly from the definition. But later we will see that in order to approximate the optimal descent direction sufficiently well we are capable of approximating f_0' sufficiently well numerically. Moreover numerical methods to compute the dual element exist. Therefore it is desirable for us that the optimal descent direction

can be computed as minus the dual of f_0', thus we desire that the optimal descent direction is characterized by being minus the dual of f_0'. So we desire that the solution of (MinC) and its dual element are unique, which is the case if X and X' are strictly convex, because then f_0' is the unique solution of a strictly convex function on a convex set and X strictly convex implies that the dual mapping is well defined.

Definition 1.38 We call a Banach space X strictly convex if the norm is strictly convex. In the case that X is strictly convex, we define the **dual mapping**

$$j : (X', \|\cdot\|_{X'}) \setminus \{0\} \to S_X(0,1) \subset (X, \|\cdot\|_X)$$

implicitly in the way that for all $x' \in X' \setminus \{0\}$ we have $x'(j(x')) = \|x'\|_{X'}$.

We summarize:

Proposition 1.39 (existence, uniqueness and charact. at a Point) *Let X be reflexive Banach space, such that X and X' are strictly convex, i.e. they have strictly convex norms, and $x \in X$. Assume further $f : X \to \mathbb{R}$ is locally Lipschitz continuous.*
Then there exists exactly one $f_0' \in \partial f(x)$ with

$$\|f_0'\| = \min\left\{\|f'\| \mid f' \in \partial f(y)\right\} .$$

If $0 \notin \partial f(x)$ there exists exactly one $d_0 \in S_X(0,1)$ with

$$\langle f_0' \,|\, d_0\rangle = -\|f_0'\|$$

and exactly one $d_1 \in S_X(0,1)$ with

$$f^0(x; d_1) = \min_{\|d\|=1} f^0(x; d).$$

Further $d_1 = d_0$ and d_0 is the optimal descent direction of f in x and

$$f^0(x; d_0) = \langle f_0' \,|\, d_0\rangle = -\|f_0'\| .$$

Proof. Proposition 1.31 says that f_0', d_0 and d_1 exist. Since X and X' are strictly convex, f_0' and d_0 are unique. Proposition 1.31 gives now that $d_0 = d_1$. ♢

Remark 1.40

1. This means that in this case the optimal descent direction is characterized by being minus the dual element of f_0'.

2. If X and X' are strictly convex and reflexive, then the dual mapping is continuous, cf. I. Cioranescu, [12, Chapter II, Prop. 5.5].

3. The optimal descent direction of Proposition 1.39 is stable, in the sense that there exists a neighborhood $U(d_0)$ of d_0 such that every element of $U(d_0)$ is a descent direction too. To see this, we observe that $f^0(x;\cdot)$ is Lipschitz continuous, which means that for every direction d in a neighborhood of d_0 the directional derivative $f^0(x;d)$ is negative and therefore d is also a descent direction. Further $f^0(\cdot;\cdot)$ is upper semicontinuous, which gives us that all d in a neighborhood $U(x)$ of x are descent directions of f on this neighborhood $U(x)$.

4. The choice of d_0 in Proposition 1.39 depends on the choice of the norm on X. If we consider for example $f : \mathbb{R}^2 \to \mathbb{R}$, $(x,y) \mapsto x^2 + y^2$ and the Euclidean norm and the norm $\|(x,y)\|_*^2 := \frac{x^2}{4} + \frac{7y^2}{4}$ and the point $x := \frac{1}{\sqrt{2}}(1,1)$. Then we gain, depending on the norm, two different d_0! For the Euclidean norm $d_0 = (1,1)$ and for the other norm $d_0 = \left(\sqrt{4}, \sqrt{\frac{4}{7}}\right)$.

We point out that not only finding the optimal norm for a specific problem and given the norm finding the optimal descent direction are very difficult. Even finding numerically a descent direction at all is often difficult. Creating an algorithm to find a descent direction at all is a huge success of this work and has not been done so far in this generality up to our knowlege. In order to find numerically a descent direction at all, which will be an approximation of the optimal descent direction, we can say that the assumption, X and X' are strictly convex, is not such a large restriction in the sense that it holds:

Lemma 1.41 *In a reflexive Banach space X there exists an equivalent norm, such that X' and X are strictly convex.*

Proof. The proof is given in I. Cioranescu [12, Theorem III.2.9]. ♢

However we do not know and do not discuss in which way changing the norm impacts the behaviour of the algorithms. E.g. varying the norm might change drastically the efficience of the algorithm to compute the dual element. But is good to know that at least theoretically we can switch from a not strictly convex norm to one that is and obtain at least a descent direction.

1.3 The Gradient on a Set

From the analytical point of view the generalized gradient at point x is a very fertile concept, which leads to various applications and the optimal descent direction can be used to prove

several existence results. However we are mainly interested in numerical applications. There we have two big problems. We can often not compute the generalized gradient and we do not know how to approximate the optimal descent direction. But even if in situations where we know the generalized gradient, it might not help. In the situation of the function $f(x,y) := |x| + \varepsilon|y|$, which we considered in Figure 1, we know the generalized gradient at each point, which is the set containing only the derivative at (x,y) for $(x,y) \in (\mathbb{R} \setminus \{0\})^2$. So if we combine the steepest descent direction with a linesearch, we gain nothing since with probability 1 after each iteration the resulting point is not an element of the axes. So we gain the same oszillating behaviour with probability 1. Another typical example is the Rosenberg function

$$f(x) = \sum_{i=0}^{N} (1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2 \ .$$

It is well know that steepest descent directions show very bad behaviour in the sense that we observe again strong oscillation. It is well known that Newton type methods give much better results for the Rosenberg function. The idea of Newton type methods is to linearize the gradient and to predict the gradients (These are essentially the optimal ascent directions.) on a small neighborhood this way. Of course for higher dimensions N computing the second derivative is very expensive, which is why one often uses just Newton type methods. These collect some gradients on a neighborhood and use them to approximate the Hessian at least in the essential directions. Formally those methods do not require that the function is two times differentiable. Numerically those algorithms do not know what the function looks like except for the points where we compute the gradients. So it does not matter whether the function is smooth, or nonsmooth or if we consider a smooth approximation of the nonsmooth function which is exact in those points, where we compute the gradient. The algorithms always look the same. This tells us that we have to develope a theory which does not distinguish between smooth and nonsmooth functions. It should not matter whether the function is smooth and has high curvatures in some directions or wether it is a nonsmooth function, since approximating nonsmooth functions by smooth functions results in high curvatures of the approximating function close to the points where the function is nonsmooth. Therefore it does not make sense to use tools for smooth functions like the Hessian matrix or approximations of it. But the good behaviour of the Newton type methods for the Rosenberg function and other functions suggests to consider the gradients on a neighborhood of our iteration point. This is also what the function $f(x,y) := |x| + \varepsilon|y|$ suggests. There we would like to make a "descent step on a neighborhood" of our iteration point, to avoid the oscillation. But therefore we need to know somehow the gradients in a neighborhood of the iteration points. This will be the aim in the following. First we present an analytical theory for optimal descent directions on a set and gradients on a set, although we can

never compute them numerically, because the understanding we gain there helps us to create numerically implementable algorithms later, just like understanding the Newton method helps to understand Newton type methods. When it comes to implementing algorithms, we will be satisfied by approximating those analytical tools.
Following the ideas of Clarke, the gradient on a set should at least contain all elements of the generalized gradients at each point of the set, thus in the smooth case it should contain the gradient of each point of the set. Moreover to gain existence results of minimal elements and to avoid technical difficulties one has to demand that the gradient on a set is convex and in some sense compact as it is the case in the Clarke calculus. Similar the derivative in some direction on a set A should at least take into account the generalized derivatives in that direction at each point of A. Further we desire that if the derivative at A in some direction is negative we get that the generalized derivative in this direction is negative in each point of A too, because this implies that this direction is a descent direction for every point of A. This leads naturally to the following definition.

Definition 1.42
Let $f : X \to \mathbb{R}$ be Lipschitz continuous on a neighborhood of $A \subset X$.

1. The **gradient of f at the set** $A \subset X$ is defined as

$$\partial^{conv} f(A) := \overline{conv}^* \bigcup_{y \in A} \partial f(y) \subset X' , \tag{1.43}$$

where we consider the weak*-closure in X'.[2] In the case[3] that $A = \overline{B_X(x, \varepsilon)}$ for some $x \in X$ and $\varepsilon > 0$ we simply write

$$\partial^{\varepsilon} f(x) := \partial^{conv} f(\overline{B_X(x, \varepsilon)}) = \overline{conv}^* \bigcup_{y \in \overline{B_X(x, \varepsilon)}} \partial f(x)$$

and call it the **gradient of f at the $\varepsilon-$neighborhood of** x.

[2] We mention that since we consider the weak*-closure of a convex set, we could also take the closure with respect to the norm in the dual space by Mazur's lemma in the case that X is reflexive.

[3] The reason why we take here the closure is of purely technical nature. To write down the later Algorithm 3.3 as simple as we will do, we need that the gradients on the boundary also belong to $\partial^{\varepsilon} f(x)$ to get rid of technical difficulties. We could also define $\partial^{\varepsilon} f(x)$ for the open ball and would loose no results. It would be just more technical effort. It is like the question, wether a neighborhood is open itself or just contains an open set.

2. We define[4] the directional derivative at A in direction $d \in X$ by

$$f_A^0(d) := \sup_{x \in A} f^0(x;d) = \sup_{x \in A} \limsup_{y \to x,\ t \downarrow 0} \frac{f(y+td)-f(y)}{t} .$$

We use the abbreviation

$$f_\varepsilon^0(x;d) := f^0_{\overline{B_X(x\,,\,\varepsilon)}}(d) .$$

3. In the case $0 \notin \partial^{conv} f\,(A)$ we call $d_0 \in S_X\,(0,1)$ a **steepest** or **optimal descent direction of f at $A \subset X$ related to the norm $\|\cdot\|$ of X**,[5] if d_0 is a descent direction of f on A and minimizes the directional derivative on A, thus (cf. Definition 1.29)

$$f_A^0(d_0) = \inf_{\|d\| \le 1} f_A^0(d) .$$

Remark 1.44

- By definition we obtain that $f^0(x;d) = f_A^0(d)$ and $\partial^{conv} f\,(A) = \partial f(x)$ in the case $A = \{x\}$.
- It follows at once by Proposition 1.6 that if f is Lipschitz continuous on a neighborhood of A, then $\partial^{conv} f\,(A)$ is convex, bounded and weak*-closed, which means that it is weak*-compact.
- A. Goldstein defined the so called $\varepsilon-$generalized gradient of x in [24] for a finite dimensional space X. His definition was (using our notation):

  Definition 1.45 Let $\hat{B_X}[x,\varepsilon] := \left\{ y \in \overline{B_X\,(0\,,\,\varepsilon)} \;\middle|\; \nabla f(y) \text{ exists} \right\}$ and let $(\vartheta_k)_{k\in\mathbb{N}}$ be any sequence of positive numbers converging downwards to 0.
  The $\varepsilon-$generalized gradient at x is the set

$$\delta_\varepsilon f(x) = conv \bigcap_{k=1}^{\infty} \overline{\left\{ \nabla f(y) \;\middle|\; y \in \hat{B_X}[0, \varepsilon + \vartheta_k] \right\}} . \tag{1.46}$$

  This definition is of course motivated by Proposition 1.7 and Proposition 1.20. If X is finite dimensional and $A = \overline{B_X\,(0\,,\,\varepsilon)}$, one easily sees that (1.43) and (1.46) are equivalent. But our definition works also, if X is not finite dimensional and the definition is a bit easier to remember, at least in our opinion.

[4] We use the notation $f_A^0(d)$ to avoid confusion with the set

$$f^0(A;d) := \left\{ f^0(x;d) \;\middle|\; x \in A \right\} .$$

[5] If the norm is clear from the context we omit the "related to the norm $\|\cdot\|$ of X".

- A. Goldstein used the notation $\delta f(x)$ for the generalized gradient and $\delta_\varepsilon f(x)$ for the $\varepsilon-$generalized gradient. δ is seldomly used notation nowadays, so we take ∂ as suggested by Clarke in [13]. We also do not use the notation $\partial_\varepsilon f(x)$, because this notation is common for the $\varepsilon-$subdifferential[6] as used e.g. by W. Alt in [1, 2] and neither is the $\varepsilon-$subdifferential a generalization of the generalized gradient of f at the set $B[x,\varepsilon]$, nor the other way around. We can not generalize Proposition 1.8. This is also, why we do not use the term "$\varepsilon-$generalized gradient", because up to our experience, this leads to confusion for people familiar with the concept of the $\varepsilon-$subdifferential. The nomenclature "generalized gradient of f at the $\varepsilon-$neighborhood of x" underlines better that we consider all gradients around x and not some pertubation as in the $\varepsilon-$subdifferential.

We can generalize naturally Proposition 1.23.

Proposition 1.47 (sufficient condition)
Let X be a Banach space and $f : X \to \mathbb{R}$ be Lipschitz continuos on A. If $f_A^0(d_0) < 0$, then d_0 is a descent direction on A.

Proof. This follows directly from the definition and the sufficient condition at a point, cf. Proposition 1.23. ♢

Example 1.48 *We come back to the initial function $f(x,y) := |x| + \alpha|y|$ with $0 < \alpha << 1$, where the steepest descent method was oscillating as we have seen in Figure 1. Assume again $(x,y) \approx (0,10)$ and $1 > \varepsilon > |x|$. If we consider on $\mathbb{R}^2$ the Euclidian norm we find*

$$\partial^\varepsilon f(x,y) = \left\{(s,\alpha) \,\middle|\, s \in [0,1]\right\} .$$

The vector $f_0' = (0,\alpha)$ has the smallest norm of all vectors in $\partial^\varepsilon f(x,y)$. Minus its dual element is $d_0 = (0,-1)$, which is a descent direction on the entire $\mathbb{R}\times\mathbb{R}_{>0}$. If we do now a line-search in direction d_0, we get directly close to the minimizer $(0,0)$ and avoid calculating all the gradients in Figure 1 by calculating just one gradients on a set.[7]

Next we show that generalizing directly the Definition 1.4 of Clarke would lead to the same definition.

Lemma 1.49 *Let $f : X \to \mathbb{R}$ be Lipschitz continuous on a neighborhood of $A \subset X$. Then*

$$f_A^0(d) \;=\; \sup\left\{\left\langle f' \,\middle|\, d\right\rangle \;\middle|\; \exists x \in A \,:\, f' \in \partial f(x)\right\}$$

[6] For a convex function $f : X \to \mathbb{R}$ we define the ε-subdifferential in the point x as the set $\partial_\varepsilon f(x) := \left\{f' \in X' \,\middle|\, \forall d \in X \,:\, f(x+d) \geq f(x) + \left\langle f' \,\middle|\, d\right\rangle - \varepsilon\right\}$.

[7] We mention that if we would have chosen the $\varepsilon-$subgradient, we would have gained the same result.

$$= \max\left\{\left\langle f' \,\middle|\, d\right\rangle \;\middle|\; f' \in \partial^{conv} f\,(A)\right\}$$

and

$$\partial^{conv} f\,(A) = \left\{f' \in X' \,\middle|\, \forall x \in X \,:\, \left\langle f' \,\middle|\, x\right\rangle \leq f_A^0(d)\right\} \;. \tag{1.50}$$

Proof. Proposition 1.6 implies the first equality. With the knowledge that linear functions defined on convex sets attain their extrema on the extreme points of the convex set, we calculate

$$\begin{aligned} f_A^0(d) &= \sup_{f' \in \bigcup\limits_{x\in A} \partial f(x)} \left\langle f' \,\middle|\, d\right\rangle \\ &= \sup_{f' \in \mathrm{conv} \bigcup\limits_{x\in A} \partial f(x)} \left\langle f' \,\middle|\, d\right\rangle \\ &= \sup_{f' \in \overline{\mathrm{conv}}^{*} \bigcup\limits_{x\in A} \partial f(x)} \left\langle f' \,\middle|\, d\right\rangle \\ &= \sup_{f' \in \partial^{conv} f(A)} \left\langle f' \,\middle|\, d\right\rangle \\ &= \max_{f' \in \partial^{conv} f(A)} \left\langle f' \,\middle|\, d\right\rangle \;, \end{aligned}$$

where the last equation follows from compactness of the gradient. The equation (1.50) follows directly from [13, Propostion 2.1.4.]; which is proved in [26]; and the proved equation. ♢

Again we find in the case of a reflexive Banach space that an optimal descent direction exists and if additionally X and X' are strictly convex, it is characterized as minus the dual of the smallest element of $\partial^{conv} f\,(A)$.

Proposition 1.51 (existence of optimal descent directions)
Let X be and $f : X \to \mathbb{R}$ be Lipschitz continuous on an open set $B \subset X$. Then for all $A \subset B$ hold

1. $$\inf_{\|d\|\leq 1} f_A^0(d) = -\inf_{f' \in \partial^{conv} f(A)} \left\|f'\right\| \;. \tag{1.52}$$

2. *For every pair $(f_0', d_0) \in \partial^{conv} f\,(A) \times \overline{B_X\,(0\,,\,1)}$ such that*

$$\left\|f_0'\right\| = \min_{f' \in \partial^{conv} f(A)} \left\|f'\right\| \quad \textit{and} \quad f^0(x; d_0) = \inf_{\|d\|\leq 1} f_A^0(d) \tag{1.53}$$

holds

$$-\left\|f_0'\right\| = \left\langle f_0' \,\middle|\, d_0\right\rangle = f_A^0(d_0) = \sup_{f' \in \partial^{conv} f(A)} \left\langle f' \,\middle|\, d_0\right\rangle \;. \tag{1.54}$$

3. If X is reflexive then a pair $(f_0', d_0) \in \partial^{conv} f(A) \times \overline{B_X(0\,,\,1)}$ satisfying (1.53) exists.

PROOF. The proof is analog to Proposition 1.31 with $\partial^{conv} f(A)$ instead of $\partial f(x)$. ♢

Corollary 1.55 *Let X be a Banach space and $f : X \to \mathbb{R}$ be Lipschitz continuous on an open set $B \subset X$. Then for all $A \subset B$ hold*

$$\inf_{\|d\| \leq 1} f_A^0(d) < 0 \Leftrightarrow 0 \notin \partial^{conv} f(A)$$

and in this case for every pair $(f_0', d_0) \in \partial^{conv} f(A) \times \overline{B_X(0\,,\,1)}$ satisfying (1.33) we have d_0 is an optimal descent direction of f at A.

Theorem 1.56 (existence, uniqueness and characterization) *Let X be reflexive Banach space, such that X and X' are strictly convex, thus they have strictly convex norms, and let $f : X \to \mathbb{R}$ be Lipschitz continuous on an open set $B \subset X$. Then for all $A \subset B$ we have:*
There exists exactly one $f_0' \in \partial^{conv} f(A)$ with

$$\left\|f_0'\right\| = \min\left\{\left\|f'\right\| \;\middle|\; f' \in \partial^{conv} f(A)\right\} .$$

If $0 \notin \partial^{conv} f(A)$ there exists exactly one $d_1 \in S_X(0,1)$ with

$$\left\langle f_0' \,\middle|\, d_1 \right\rangle = -\left\|f_0'\right\|$$

and exactly one $d_0 \in S_X(0,1)$ with

$$f_A^0(d_0) = \min_{\|d\|=1} f_A^0(d).$$

Further we have $d_1 = d_0$ and d_0 is the optimal descent direction of f at A and

$$f_A^0(d_0) = \left\langle f_0' \,\middle|\, d_0 \right\rangle = -\left\|f_0'\right\| .$$

PROOF. The proof goes analog to Proposition 1.39. ♢

Remark 1.57

- In the case $0 \notin \partial^{conv} f(A)$ this means the optimal descent direction is characterized by being minus the dual of f_0' and it is characterized by being a saddle point of the min-max problem. This characterization is very important for us, because clearly the optimal descent direction is the direction we want to compute from the analytical point of view.

But numerically this is nearly impossible. But approximating the smallest gradient f_0' and its dual element, is something we can do very well. This is also what we will do later.

- It is important to us that we consider the pair (f_0', d_0), since we are in a Banach space. They replace the optimal descent direction in the situation of a smooth function defined on a Hilbert space. It is a step we have to make to gain comparable powerful theory as in the Hilbert space.

- We mention here that Propsition 1.39 is basically well known in convex analysis. Especially if f is convex, Propsition 1.39 is common knowledge. But the definition of an optimal descent direction on a set and with it Proposition 1.51 and Theorem 1.56 are new concepts, which we could not find in the literature. But there are very similar results one can find for $\varepsilon-$subdifferentials. Further some partial results of Proposition 1.51 and Theorem 1.56 can be found in the work by Goldstein [24] for finite dimensional X, but the reference is missing and they are not proved there.

- We see that f_0' plays a crucial role in our theory. We will later try to approximate f_0' by the element $\arg\min\left\{\|f'\| \;\middle|\; f' \in B \subset \partial^{conv} f(A)\right\}$ for suitable convex B. To solve this minimization problem, we will later choose the norm of X such that we can compute easily the minimizer and the dual element, which gives us an approximation of the optimal descent direction. We will even allow the algorithm to choose a different norm in each iteration step. Normally we choose Hilbert space norms. But for the general theory it is not important which norm we take, as long as the Banach space is reflexive and the norm and the dual norm are strictly convex. And we have to keep in mind that the optimal descent direction depends on the norm.

Next we show that the optimal descent direction is stable in the sense that if $\|f'\|$ is sufficiently close to the norm of the minimizer, then minus the dual element of f' is a descent direction on $\overline{B_X(x\,,\,\varepsilon)}$. We simplify and generalize a result from [4, Lemma 3.1] and [36, Lemma 3.1] and interpret it in our setting.[8]

Lemma 1.58
Let X be a reflexive uniformly convex Banach space or finite dimensional and strictly convex. Further we assume that X' is also strictly convex.
Let $0 \notin C' \neq \emptyset$ be a weak-compact, convex subset of X' and $\delta \in]0,1[$. Then there exists some $\tau > 0$ such that*

$$c_0' \in C' \quad \textit{and} \quad \left\|c_0'\right\| \leq \inf\left\{\left\|c'\right\| \;\middle|\; c' \in C'\right\} + \tau$$

[8] [4, Lemma 3.1] and [36, Lemma 3.1] were only formulated for the $\mathbb{R}^n$ equipped with the Euclidean norm.

imply $\langle c' \,|\, j(c'_0)\rangle > \delta \|c'_0\|$ *for all* $c' \in C'$.

Corollary 1.59 (sufficient condition for descent direction)
Let X *be a reflexive uniformly convex Banach space or finite dimensional and strictly convex. Further we assume that* X' *is also strictly convex.*
Let $f : X \to \mathbb{R}$ *be Lipschitz continuous on a neighborhood of* $\overline{B_X(x\,,\,\varepsilon)}$ *and* $0 \notin \partial^\varepsilon f(x)$. *Then there exists some* $\tau > 0$ *such that for every* $f'_0 \in \partial^\varepsilon f(x)$ *with*

$$\|f'_0\| \leq \min\left\{\|f'\| \;\middle|\; f' \in \partial^\varepsilon f(x)\right\} + \tau,$$

minus the dual element $-j(f'_0)$ *of* f'_0 *is a descent direction at* $\overline{B_X(x\,,\,\varepsilon)}$ *and for every* $f' \in \partial^\varepsilon f(x)$ *we have*

$$\langle f' \,|\, j(f'_0)\rangle > \delta \|f'_0\| \ .$$

Proof. Apply Lemma 1.58 to $C' := \partial^\varepsilon f(x)$ and $f'_0 = c'_0$. ♢

Proof Lemma 1.58. We define $\inf\left\{\|c'\| \;\middle|\; c' \in C'\right\} = \alpha$. Since $0 \notin C'$ we obtain that the j is well defined on C'. If the assertion would be false, we could pick two sequences $(c'_{j,i})_{i\in\mathbb{N}} \in (C')^{\mathbb{N}\setminus\{0\}}$ with $j \in \{0,1\}$ satisfying $\left\|c'_{0,i}\right\| \leq \alpha + \frac{1}{i}$ and $\left\langle c'_{1,i} \,\middle|\, j(c'_{0,i})\right\rangle \leq \delta \left\|c'_{0,i}\right\|$. By weak*-compactness of C' we may assume $c'_{j,i} \rightharpoonup^* c'_j \in C'$ for $j \in \{0,1\}$. Since $\alpha \leq \|c'_0\| \leq \lim\limits_{i\to\infty} \left\|c'_{0,i}\right\| = \alpha$, thus $\left\|c'_{0,i}\right\| \to \|c'_0\|$, and X is assumed uniformly convex or finite dimensional, it follows by [12, Chapter II, Prop. 2.8] that $c'_{0,i} \to c'_0$. [12, Chapter II, Prop. 5.5] gives that the duality mapping is continuous. Thus

$$\langle c'_1 \,|\, j(c'_0)\rangle \leq \delta \|c'_0\| \,. \tag{1.60}$$

We apply the Duality Theorem [60, Theorem 49b] to the sets C' and $\overline{B_X(0\,,\,1)}$ and the function $L(c', d) := \langle c' \,|\, d\rangle$ and gain some $\tilde{c}' \in C'$ and $\tilde{d} \in \overline{B_X(0\,,\,1)}$ with

$$\begin{aligned}\sup_{d\in\overline{B_X(0\,,\,1)}} \left\langle \tilde{c}' \,\middle|\, d\right\rangle &= \inf_{c'\in C'} \sup_{d\in\overline{B_X(0\,,\,1)}} \langle c' \,|\, d\rangle = \left\langle \tilde{c}' \,\middle|\, \tilde{d}\right\rangle \\ &= \sup_{d\in\overline{B_X(0\,,\,1)}} \inf_{c'\in C'} \langle c' \,|\, d\rangle = \inf_{c'\in C'} \left\langle c' \,\middle|\, \tilde{d}\right\rangle\end{aligned}$$

and so

$$\|\tilde{c}'\| = \sup_{d\in\overline{B_X(0\,,\,1)}} \langle \tilde{c}' \,|\, d\rangle = \inf_{c'\in C'} \sup_{d\in\overline{B_X(0\,,\,1)}} \langle c' \,|\, d\rangle = \inf_{c'\in C'} \|c'\| \ , \tag{1.61}$$

$$\|\tilde{c}'\| = \inf_{c'\in C'} \sup_{d\in\overline{B_X(0\,,\,1)}} \langle c' \,|\, d\rangle = \left\langle \tilde{c}' \,\middle|\, \tilde{d}\right\rangle \ , \tag{1.62}$$

$$\left\|\tilde{c}'\right\| = \sup_{d\in\overline{B_X(0\,,1)}} \inf_{c'\in C'} \left\langle c' \,\middle|\, d\right\rangle = \inf_{c'\in C'} \left\langle c' \,\middle|\, \tilde{d}\right\rangle \ . \tag{1.63}$$

Since X and X' are strictly convex $\tilde{c}' = c_0'$ by (1.61) and $\tilde{d} = j(\tilde{c}') = j(c_0')$ by (1.62). (1.63) gives now $\left\langle c' \,\middle|\, j(c_0')\right\rangle \geq \|c_0'\|$ for all $c' \in C'$ which is a contradiction to (1.60). ♢

The next lemma will be useful in the proofs to show convergence of our algorithm. It basically states that if x is not a critical point of f, then there exists an ε such that there exists a descent direction on $\overline{B_X\left(x\,,\,\varepsilon\right)}$ and the gradients are locally bounded away from zero.

Lemma 1.64 (bounded away from zero)
Let X be a Banach space and $f : X \to \mathbb{R}$ be locally Lipschitz continuous. We assume $0 \notin \partial f(x)$ for some $x \in X$. Then there exist $\varepsilon > 0,\ K > 0$ and $d \in S_X\left(0,1\right)$ such that we have

$$f_\varepsilon^0(x;-d) \leq -K < 0$$

or equivalently for all $f' \in \partial^\varepsilon f\left(x\right)$ hold

$$\left\|f'\right\|_{X'} \geq \left\langle f' \,\middle|\, d\right\rangle \geq K \ .$$

Thus $0 \notin \partial^\varepsilon f\left(x\right)$ and $-d$ is a descent direction on $\overline{B_X\left(x\,,\,\varepsilon\right)}$.

Proof.
Propsition 1.25 says that for every $x \in X$ with $0 \notin \partial f(x)$ there exists some $d \in S_X\left(0,1\right)$ such that

$$\inf_{f'\in\partial f(x)} \left\langle f' \,\middle|\, d\right\rangle =: 2K > 0.$$

This means that $f^0(x;-d) = -2K$. Since $f^0(\cdot;-d)$ is upper semi continuous, there exists some $\varepsilon > 0$ such that

$$\begin{aligned} \forall y \in \overline{B_X\left(x\,,\,\varepsilon\right)} :\ f^0(y;-d) \ &< -K \\ \Rightarrow \forall y \in \overline{B_X\left(x\,,\,\varepsilon\right)}\ \forall f' \in \partial f(y) :\ \left\langle f' \,\middle|\, -d\right\rangle \ &< -K \\ \Rightarrow \forall f' \in \overline{conv}^*\left(\bigcup_{y\in\overline{B_X(x\,,\varepsilon)}} \partial f(y)\right) :\ \left\langle f' \,\middle|\, -d\right\rangle \ &\leq -K \ . \end{aligned}$$

♢

2 General Aspects of Descent Algorithms

We start with a theoretical algorithm, which was given by A. Goldstein in [24]. For this we define for $q \in \mathbb{R}$, $\varepsilon > 0$

$$\begin{aligned} D(x,q,\varepsilon) &:= \left\{ d \in X \;\middle|\; \|d\| = 1 \text{ and } \forall f' \in \partial^\varepsilon f(x) : -q \geq \left\langle f' \;\middle|\; d \right\rangle \right\} , \\ &= \left\{ d \in X \;\middle|\; \|d\| = 1 \text{ and } -q \geq f_\varepsilon^0(x;d) \right\} , \end{aligned}$$

by Proposition 1.51. We observe that our optimal descent direction d_0 is in $D(x,q,\varepsilon)$ if

$$-q \geq f_\varepsilon^0(x;d_0)$$

or equivalent[9]

$$\min_{f' \in \partial^\varepsilon f(x)} \left\| f' \right\| \geq q \, .$$

With this definition the algorithm becomes:

Algorithm 2.1 (theoretical algorithm by A. Goldstein)

1. *Initialization: Choose $x_0 \in \mathbb{R}^n$ and set $k = 0$.*

2. *Repeat: Choose $\varepsilon_k > 0$ and*

$$d_k \in D\left(x_k, \frac{1}{2} \min_{f' \in \partial^{\varepsilon_k} f(x_k)} \left\| f' \right\|, \varepsilon_k \right) = D\left(x_k, -\frac{1}{2} \min_{\|d\| \leq 1} f_{\varepsilon_k}^0(x_k;d), \varepsilon_k \right) ,$$

   define $x_{k+1} = x_k + \varepsilon_k d_k$ and increment k by one.

This theoretical algorithm converges if we choose ε_k proper and if additionally $\dim X < \infty$.

Lemma 2.2 (convergence of the algorithm by A. Goldstein)
Lef $f : X \to \mathbb{R}$ be Lipschitz continuous. Assume $X = \mathbb{R}^n$ and define $Z = \left\{ x \in X \;\middle|\; 0 \in \partial f(x) \right\}$. If we choose x_k according to Algorithm 2.1 and ε_k such that

$$\varepsilon_k \downarrow 0$$

and

$$\sum_{k \in \mathbb{N}} \varepsilon_k = \infty$$

[9] We recall $f_\varepsilon^0(x;d_0) = -\left\| f_0' \right\| = -\min\left\{ \|f'\| \;\middle|\; f' \in \partial^\varepsilon f(x) \right\} \geq q$.

we obtain

$$dist(x_k, Z) \to 0$$

as $k \to 0$.

PROOF. The proof can be found in A. Goldstein [24]. ♢

Remark 2.3 The problem with this algorithm is of course that in general $\frac{1}{2} \min_{f' \in \partial^\varepsilon f(x_k)} \|f'\|_{X'}$ and $\partial^\varepsilon f(x_k)$ are unknown. This problem was not solved by A. Goldstein in [24].

Our goal in the proceeding will be to approximate $\frac{1}{2} \min_{f' \in \partial^\varepsilon f(x_k)} \|f'\|_{X'}$ and $\partial^\varepsilon f(x_k)$ sufficiently good.

2.1 Efficient Step Sizes

Before we can deal with a sufficiently good approximation, we recall the idea of efficient step sizes, because we need them to define a good approximation. In the literature one can find a well developed theory about efficient step sizes. We will only give the definition, motivate them and give some simple results. We follow the ideas of the introduction by W.Alt, [1].

For simplicity of the notation and to cite the results of the introduction [1] we assume that X is a Hilbert space and $f \in C^1(X)$ as long as we are talking about efficient step sizes. We deal with the question: How do we choose the step size such that the convergence of the sequence $(x_k)_{k \in \mathbb{N}}$ produced by a descent algorithm implies that it converges to a critical point. We use the notation $x_{k+1} =: x_k + \sigma_k d_k$, where $\sigma_k > 0$ is the step size and d_k is the normalized descent direction, thus $\|d_k\| = 1$. A sufficient condition would be of course

$$\nabla f(x_k) \to 0 \text{ as } k \to \infty \, .$$

This is a harsh condition. Therefore we want to ensure first

$$\left\langle \nabla f(x_k) \,\middle|\, d_k \right\rangle \to 0 \text{ as } k \to \infty \tag{2.4}$$

by a proper choice of the step size. For small σ_k it is

$$f(x_k + \sigma_k d_k) - f(x_k) \approx \sigma_k \left\langle \nabla f(x_k) \,\middle|\, d_k \right\rangle \, .$$

Since we use a descent method, the left hand side converges to 0 by assumption in the case that f is bounded from below. To gain that also the right hand side converges to 0, we demand

$$f(x_k + \sigma_k d_k) - f(x_k) \leq c_1 \sigma_k \left\langle \nabla f(x_k) \,\middle|\, d_k \right\rangle \leq 0 \tag{2.5}$$

for some constant $c_1 > 0$. But to gain (2.4) we have to ensure that σ_k does not converge to 0 too fast. This motivates the sufficient condition

$$\sigma_k \geq -c_2 \left\langle \nabla f(x_k) \,\middle|\, d_k \right\rangle \tag{2.6}$$

for some constant $c_2 > 0$. We wish to say that a step size σ_k satisfies the principle of sufficient descent, if the conditions (2.5) and (2.6) hold, as suggested by P. Spellucci in [55]. But of course we can only define a sufficient descent strategy, since for a single point and any step size we can always find such constants.

Definition 2.7 A **step size strategy**[10] for a locally Lipschitz continuous function f is a mapping

$$\sigma : X \times S_X(0,1) \to \mathbb{R}_{\geq 0}$$

such that

$$f^0(x;d) \geq 0 \;\; \Rightarrow \sigma_{xd} = 0 \,.$$

The evaluation $\sigma(x,d)$ is called step size at x in direction d. A step size strategy σ satisfies the **principle of sufficient descent** for f, if

$$f(x + \sigma_{xd} d) - f(x) \leq c_1 \sigma_{xd} \, f^0(x;d)$$

and

$$\sigma_{xd} \geq -c_2 \, f^0(x;d)$$

for some constants c_1, $c_2 > 0$. A step size strategy is called **efficient** for f, if for some constant $c > 0$ we have that $\sigma_{xd} \neq 0$ implies

$$f(x + \sigma_{xd} d) - f(x) \leq -c \, f^0(x;d)^2 \,. \tag{2.8}$$

If a step size strategy satisfies the principle of sufficient descent, then the step size strategy is efficient with $c = c_1 c_2$. We mention an existence result in the smooth case without proof.

Proposition 2.9 *Let $f \in C^1(X, \mathbb{R})$ such that ∇f is Lipschitz continuous and let $X = \mathbb{R}^n$ be a*

[10] we use the abbreviation $\sigma_{xd} = \sigma(x,d)$

Hilbert space. Then every mapping $d : X \to S_X(0,1)$ admits an efficient step size strategy, thus for some $c > 0$ and $\sigma : X \times S_X(0,1) \to \mathbb{R}_{\geq 0}$ holds (2.8) with σ_{xd} replace by $\sigma_{xd(x)}$.

PROOF. If $f^0(x;d) = \langle \nabla f(x) \,|\, d(x) \rangle < 0$, this is a direct consequence of [1, Lemma 4.4.3], else we simply choose $\sigma_{xd} = 0$. ♢

Remark 2.10 A comparable proposition is not true in the nonsmooth case. Take the absolute value in $\mathbb{R}$ as function. Then for all fixed $c > 0$ we can not find σ such that (2.8) holds if we take $\|x\| \neq 0$ sufficiently small and $d := -\frac{x}{\|x\|}$.

Of course Assumption (2.4) alone does not imply that $\nabla f(x_n) \to 0$. For this we demand that d_k is chosen properly. Typically one demands that d_k is chosen gradient related in x_k, cf. [1].

Definition 2.11 Let $d : X \to S_X(0,1)$ and $f : X \to \mathbb{R}$ be differentiable. The mapping d is called **gradient related** for f, if

$$\left(-f^0(x;d(x)) =\right) - \langle \nabla f(x) \,|\, d(x) \rangle \geq c_3 \|\nabla f(x)\|$$

for some constant $c_3 > 0$.

One easily proves:

Proposition 2.12 *Assume that $f : X = \mathbb{R}^n \to \mathbb{R}$ is bounded from below and differentiable, $d : X \to S_X(0,1)$ is gradient related, $\sigma : X^2 \to \mathbb{R}_{\geq 0}$ is efficient and $x_{k+1} = x_k + \sigma_k d_k$, where $d_k := d(x_k)$ and $\sigma_k = \sigma(x_k, d_k)$. Then*

$$\nabla f(x_k) \to 0 \text{ for } k \to \infty \,.$$

Now we define a generalization of gradient related to locally Lipschitz continuous functions.

Definition 2.13 Let $f : X \to \mathbb{R}$ be locally Lipschitz continuous. A mapping $d : X \to S_X(0,1)$ is called

1. **gradient related**, if there exists a constant $c_3 \in]0,1[$ such that for all $x \in X$ holds

$$\left(\min_{f' \in \partial f(x)} - \langle f' \,|\, d(x) \rangle = \right) - f^0(x;d(x)) \geq c_3 \min_{f' \in \partial f(x)} \|f'\| \quad .$$

2. and $\varepsilon-$**gradient related**, if there exists a constant $c_3 \in]0,1[$ such that for all $x \in X$ holds

$$\left(\min_{f' \in \partial^\varepsilon f(x)} - \langle f' \,|\, d(x) \rangle = \right) - f^0_\varepsilon(x;d(x)) \geq c_3 \min_{f' \in \partial^\varepsilon f(x)} \|f'\| \quad .$$

Remark 2.14 For a smooth function the classical gradient related direction is minus the dual of the gradient. In the nonsmooth case we would choose minus the dual of the norm-minimizing element of $\partial^\varepsilon f(x)$ (or a sufficient good approximation) as optimal descent direction d_0. By Theorem 1.56 we would obtain[11] in the case $0 \notin \partial^\varepsilon f(x)$

$$\min_{f' \in \partial^\varepsilon f(x)} \langle f' \mid -d_0 \rangle = \langle f_0' \mid -d_0 \rangle = \|f_0'\| = \min_{f' \in \partial^\varepsilon f(x)} \|f'\| \ ,$$

which says that d_0 is ε-gradient related. In practice we will not guarantee gradient related, since we do not know the value $\min_{f' \in \partial^\varepsilon f(x)} \|f'\|$, but we will ensure "convergence" of the iterations in the so called null steps, which we define later, to this gradient related direction. This will be sufficient to show that accumulation points of the sequence $(x_k)_{k \in \mathbb{N}}$ are critical points.

We finish this section by giving the most important efficient step size strategies.

Example 2.15 (Exact Step Size)
We consider the case $f \in C^1(X, \mathbb{R})$. Suppose we can choose σ such that

$$f(x + \sigma_{xd} d) = \min_{s \geq 0} f(x + sd).$$

Then we call this σ **exact** *step size strategy and σ is efficient for f if the gradient of f is Lipschitz continuous and the level sets are compact, cf. [1, Satz* 4.5.2*].*

Example 2.16 (Step Size by Armijo)
We consider the case $f \in C^1(X, \mathbb{R})$. Suppose we can choose σ^A such that for some constants $\delta \in]0, 1[$ and $c_2 > 0$ the following inequalities hold:

$$f(x + \sigma^A_{xd} d) - f(x) \leq \delta \sigma^A_{xd} \langle \nabla f(x) \mid d \rangle \,, \tag{2.17}$$

$$\sigma^A_{xd} \geq -c_2 \langle \nabla f(x) \mid d \rangle \ . \tag{2.18}$$

This says that σ^A satisfies the principle of sufficient descent for f with $c_1 = \delta < 1$ and therefore σ^A is efficient. σ^A is called Armijo step size strategy.

Example 2.19 (Step Size by Powell)
We consider the case $f \in C^1(X, \mathbb{R})$. If we can choose σ^P such that for some constants

$$0 < \delta < \beta < 1$$

[11] We obmit the index k.

the following inequalities hold:

$$f(x + \sigma_{xd}^P d) - f(x) \leq \delta \sigma_{xd}^P \left\langle \nabla f(x) \,\middle|\, d \right\rangle, \tag{2.20}$$

$$\left\langle \nabla f(x + \sigma_{xd}^P d) \,\middle|\, d \right\rangle \geq \beta \left\langle \nabla f(x) \,\middle|\, d \right\rangle \ . \tag{2.21}$$

If ∇f is Lipschitz continuous and f is bounded from below, then σ^P is an efficient step size, cf. [1]. σ^P is called Powell step size strategy.

In the following we will use the ideas of the Armijo step size strategy, although we are not in the smooth case. We will not use a generalization of the Armijo step size strategy, but something similar to the Armijo step size strategy. The problem that arises is that we would need a generalization of $\left\langle \nabla f(x) \,\middle|\, d \right\rangle$, which we could actually compute in practice. Typically that is not the case for $f^0(x; d(x))$.

2.2 Our Basic Descent Algorithm

Next we present our descent algorithm and prove that every accumulation point is a stationary point. We assume that X is a reflexive Banach space and $f : X \to \mathbb{R}$ is a locally Lipschitz continuous function. The algorithm looks basically like this.

Algorithm 2.22 (conceptual, rough algorithm)

1. *Initialization: Set $k = 0$, choose $x_0 \in X$, $h : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ and $\delta > 0$.*
2. *Choose $\varepsilon_k > 0$ and determine some $D_k \in \partial^{\varepsilon_k} f(x_k)$ and a dual element*

   $$d_k := j(D_k) \in S_X(0, 1)$$

   such that either the Armijo type assumption

   $$f(x_k - \varepsilon_k d_k) - f(x_k) \leq -\delta \|D_k\| \varepsilon_k \tag{Armijo}$$

   or the null step assumption, which indicates $0 \in \partial^{\varepsilon_k} f(x_k)$,

   $$\|D_k\| < h(\varepsilon_k) \tag{NS}$$

   holds.
3. *If $\|D_k\| < h(\varepsilon_k)$ then restart 2 with a smaller ε_k.*
4. *Set $x_{k+1} := x_k - \sigma_k d_k$ with $\sigma_k \geq \varepsilon_k$, increment k by one and go to 2.*

Remark 2.23

1. Later we will show how to determine D_k numerically. For this we create an inner approximation A_k of $\partial^{\varepsilon_k} f(x_k)$ as the convex hull of finitely many elements in $\partial^{\varepsilon_k} f(x_k)$ first, which is possible since $\partial^{\varepsilon_k} f(x_k)$ is convex. Then we define $D_k := \arg\min \left\{ \|f'\|_{X'} \;\middle|\; f' \in A_k \right\}$. If we manage to make A_k a sufficiently good approximation, then $-d_k$ is a good approximation of the optimal descent direction on $\overline{B_X(x_k, \varepsilon_k)}$. For further details we refer to Section 3. We remind the reader of Corollary 1.59.

2. Condition (Armijo) can be seen as generalization of (2.5). But as Remark 2.10 shows, we can not hope to find an efficient step size strategy, since we are dealing with nonsmooth functions. Therefore we do not try to generalize (2.6). But since we choose $-d_k$ as descent direction, which is minus the dual of D_k, we obtain that $-d_k$ has a good chance to be ε_k-gradient related in the generalized sense. With small norm of D_k, the chance becomes better, thus roughly speaking a null step improves the chances to gain a gradient related direction too.

3. We formulated the algorithm for a fixed norm for simplicity. Of course this is not necessary. With common assumptions like uniform equivalence of the norms like e.g. in [15], one still gains a convergent algorithm. This means that one could also implement a generalized Newton method this way, by taking the norm given by $\|\cdot\|_k^2 := \langle \cdot \,|\, H(x_k)\cdot \rangle$ in every step k, where $H(x)$ is something like a "nonsmooth Hessian matrix". But this would not give the nonsmooth Newton method given by M. Ulbrich [57]. We will deal with this idea later in Chapter 4.

Of course Algorithm 2.22 is too general to converge. For this purpose we have to formulate the algorithm a bit more precise, but still theoretical. For our algorithm we need three technical test functions H, h and g. The function h is a function which we use to test whether ε_k is too large to allow a descent on the entire ball $\overline{B_X(x_k, \varepsilon_k)}$. We recall that

$$\min \left\{ \|f'\|_{X'} \;\middle|\; f' \in \partial^{\varepsilon_k} f(x_k) \right\} \leq \|D_k\|_{X'} .$$

On the other hand, we have to prevent ε_k of going too fast to 0, otherwise the sequence might not leave a ball $\overline{B_X(x_0, r)}$ for some r. For example with the choice $\varepsilon_k = \frac{r}{2^k}$ in iteration k, when we call Step 2 the first time in iteration k, we obtain that all $x_k \in B_X(x_0, r)$. But if no stationary point is in $\overline{B_X(x, r)}$, we can not expect convergence to a critical point. Often one finds the assumption $\sum_{k \in \mathbb{N}} \varepsilon_k = \infty$, such as in the work of A. Goldstein, [24]. But this often leads to very bad convergence rates as pointed out by W. Alt in[2]. Therefore we define the function

$g : \mathbb{R}^2_{>0} \to \mathbb{R}_{>0}$ taking $\|D_k\|_{X'}$ and ε_k as parameters and demand that

$$\varepsilon_{k+1} \geq g(\|D_k\|_{X'}, \varepsilon_k) \ ,$$

when Step 2 is called the first time in iteration k. One might for example think of $g(x, y) = y$, thus $\varepsilon_{k+1} \geq \varepsilon_k$ in the beginning.
Moreover for the same reason as above ε_k mustn't go too fast to 0 in Step 3. Further we need in our proof

$$\|D_k\|_{X'} \to 0 \text{ iff } \ \varepsilon_k \to 0 \ .$$

For this reason we need a function $H : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$. Typically in other algorithms with a step like Step 3 one says $\varepsilon_{k+1} = q\varepsilon_k =: H(\varepsilon_k)$ for some $q \in (0, 1)$. But in some of our applications it turned out to be better to take something that decreases ε_k slower.
We now formulate our general assumptions on H, h and g in order to gain convergence.

Assumption 2.24 (assumptions for the basic descent algorithm)

1. *$H, h : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ are nondecreasing and*
   $\lim_{t\to 0} h(t) = 0$ and $\lim_{n\to\infty} H^n(t) := H(H^{n-1}(t)) = 0$ for all $t > 0$.[12]
2. *$g : \mathbb{R}^2_{>0} \to \mathbb{R}_{>0}$ has at least one of the following properties:*
   (a) *For any sequences $(x_k)_{k\in\mathbb{N}}$, $(y_k)_{k\in\mathbb{N}}$ we have:*
   $g(x_k, y_k) \to 0$ as $k \to 0$ implies that $x_k \to 0$ as $k \to 0$.
   (b) *For all $K > 0$ there exists some y_0 such that for all $x > K$ and $y < y_0$ holds $g(x, y) \geq y$.*
   (c) *For all $x > 0$, the functions $g(x, \cdot)$ and $g(\cdot, x)$ are nondecreasing. Moreover for all $x, y > 0$ we have*
   $$\sum_{n\in\mathbb{N}} g_x^n(y) = \infty,$$
   where $g_x^{n+1}(y) := g_x^1(g_x^n(y))$ and $g_x^1(y) = g(x, y)$.

Before we continue, we give some examples for the function g and H.

Example 2.25 (examples for suitable g and H)

1. *$g = K$, where $K > 0$ is a constant, satisfies (a).*
2. *$g(x, y) = Cy$ with $C \geq 1$ satisfies (b).*

[12] This gives that $H(s) < s$ for all $s > 0$ by monotonicity. $H^0 := H$.

3. $g(x,y) = \overline{g}(x)$ *satisfies (a), where* $\overline{g} : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ *is nondecreasing and* $\lim_{t\to 0} \overline{g}(t) = 0$.

4. *If* g_1 *and* g_2 *both satisfy each (a) or (b), then also* $g := g_1 + g_2$ *and* $g := \max\{g_1, g_2\}$ *satisfy (a) or (b) respectively.*

5. *For* $C > 0$ *we have* $g(x,y) := \max\{C, y\}$ *satisfies (a) and* $g(x,y) := \min\{C, y\}$ *satisfies (b).*

6. $H(x) = \alpha x$ *satisfies (a), where* $\alpha \in]0,1[$ *or* $H(x) := \frac{x}{1+x}$, *thus* $H(\frac{1}{n}) = \frac{1}{n+1}$.

Let us now reformulate the algorithm. The following algorithm is our basic algorithm. We do not say yet how to compute the descent direction. This will be the topic in Chapter 3; in applications we will always use Algorithm 3.3 to determine the descent direction.

Algorithm 2.26 (basic descent algorithm)

1. *Initialization: Choose* h, H *and* g *satisfying Assumption 2.24,* $\delta \in]0,1[$, $x_0 \in X, \varepsilon_{0,0} > 0$ *and set* $i = k = 0$.

2. *Choose*[13] $D_{k,i} \in \partial^{\varepsilon_{k,i}} f(x_k)$ *and a dual* $d_{k,i} \in S_X(0,1)$ *of* $D_{k,i}$ *such that the descent step assumption*
$$f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) \leq -\delta \|D_{k,i}\| \varepsilon_{k,i} \tag{DSAb}$$
or the null step assumption
$$\|D_{k,i}\| < h(\varepsilon_{k,i}) \tag{NSAb}$$
is satisfied.

3. *If* $\|D_{k,i}\| < h(\varepsilon_{k,i})$ *choose* $\varepsilon_{k,i+1}$ *sufficiently smaller than* $\varepsilon_{k,i}$, *i.e.* $\varepsilon_{k,i} \to 0$ *in the case* $i \to \infty$ *for fixed* k, *but not too small, i.e.*
$$\varepsilon_{k,i+1} \geq H(\varepsilon_{k,i})\ ;$$
e.g. the choice $\varepsilon_{k,i+1} := H(\varepsilon_{k,i})$ *satisfies both conditions; increment* i *by one and go to 2.*

4. *Set* $x_{k+1} = x_k - \sigma_k d_k$ *for some* $\sigma_k \geq \varepsilon_k := \varepsilon_{k,i}$ *such that*
$$f(x_k - \sigma_k d_{k,i}) - f(x_k) \leq -\delta \|D_{k,i}\| \sigma_k \tag{DSAb2}$$

[13] If $0 \notin \partial^{\varepsilon_{k,i}} f(x_k)$ by Lebourg's Theorem and Proposition 1.51 there exist $D_{k,i}$ and $d_{k,i}$ satisfying (DSAb) for every reflexive Banach space. If $0 \in \partial^{\varepsilon_{k,i}} f(x_k)$ we have (L-NSAb). Later in Section 3 we will come to the question of finding them. In applications we will apply Algorithm 3.3 or Algorithm 3.34. There we have to demand that the Banach space is a Hilbert space or a closed subspace of a Sobolev space.

and choose

$$\varepsilon_{k+1,0} \geq g(\|D_k\|, \varepsilon_k)$$

where $D_k := D_{k,i}$*, set* $i = 0$*, increment* k *by one and go to 2.*

We will generalize and specialize this algorithm in Section 2.3. But this is essentially our algorithm we use in this work.

Remark 2.27 Instead of choosing $\varepsilon_{k+1,0} \geq g(\|D_k\|, \varepsilon_k)$, we could also choose $\varepsilon_{k+1,0}$ such that $\sum_{k\in\mathbb{N}} \varepsilon_{k,0} = \infty$.

Remark 2.28 Here we see why we need a control about choosing $\varepsilon_{k,0}$ by g. Since $\varepsilon_{k,i} \leq \varepsilon_{k,0}$, we might choose ε_k too small, such that

$$\|x_N - x_0\| \leq \sum_{k\in\mathbb{N}} \|x_{k+1} - x_k\| \leq \sum_{k\in\mathbb{N}} \varepsilon_{k,0} =: K_0 < \infty.$$

Therefore if the only critical point has a larger distance to x_0 than K_0, we will never reach it. For this reason we could also avoid the function g if we require that $\sum_{k\in\mathbb{N}} \varepsilon_{k,0} = \infty$. For the same reason we are not allowed to let $i \to \varepsilon_{k,i}$ decrease too fast in the null step, i.e. if $\|D_{k,i}\| < h(\varepsilon_{k,i})$.

Now we show that every accumulation point of the sequence produced by the algorithm is a critical point.

Theorem 2.29 (accumulation points are critical points)
Let X *be a reflexive Banach space,* $f : X \to \mathbb{R}$ *be a locally Lipschitz continuous function and* x_k *be iterations steps of Algorithm 2.26.*

1. *For fixed* $k, i \in \mathbb{N}$ *there exist* $D_{k,i} \in \partial^{\varepsilon_{k,i}} f(x_k)$ *and a dual* $d_{k,i}$ *satisfying (DSAb) or (NSAb).*
2. *If* $0 \notin \partial f(x_k)$ *for some fixed* $k \in \mathbb{N}$*, then there exists only finitely many* $i \in \mathbb{N}$ *such that (NSAb) is satisfied.*
3. *For any sequence* $(x_k)_{k\in\mathbb{N}} \in X^{\mathbb{N}}$ *produced by Algorithm 2.26 we have that* $(f(x_k))_{k\in\mathbb{N}}$ *is strictly decreasing and for every accumulation point* x *of* $(x_k)_{k\in\mathbb{N}}$ *it hold* $0 \in \partial f(x)$ *and* $f(x_k) \to f(x)$ *as* $k \to \infty$*.*

Proof.

1. We fix for a moment $k, i \in \mathbb{N}$. If $0 \in \partial^{\varepsilon_{k,i}} f(x_k)$ then $D_{k,i} = 0$ satisfies (NSAb) and if not we apply Proposition 1.51 and choose $D_{k,i} := f_0'$ and $d_{k,i} := -d_0$. With Lebourg's Theorem we compute for some $f' \in \partial^{\varepsilon_{k,i}} f(x_k)$

$$\begin{aligned} f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) &= \varepsilon_{k,i} \left\langle f' \,\middle|\, -d_{k,i} \right\rangle \\ &\leq -\varepsilon_{k,i} \|D_{k,i}\| < -\delta \varepsilon_{k,i} \|D_{k,i}\| \ , \end{aligned}$$

thus (DSAb) is satisfied.

2. We consider a fixed k and assume $0 \notin \partial f(x_k)$. As pointed out in Lemma 1.64 we can find ε, $K > 0$ such that for all $f' \in \partial^{\varepsilon} f(x_k)$ we have $\|f'\| > K$. Assume we would make infinitely many null steps. Since $\varepsilon_{k,i} \to 0$ as $i \to \infty$ we obtain that

$$K \leq \|D_{k,i}\| \leq h(\varepsilon_{k,i})$$

can only hold for finitely many $i \in \mathbb{N}$, since $h(\varepsilon_{k,i}) \to 0$ by Assumption 2.24, which is a contradiction to the assumption that we make infinitely many null steps.

3. $(f(x_k))_{k\in\mathbb{N}}$ is strictly decreasing by construction. Assume x to be an accumulation point of $(x_k)_{k\in\mathbb{N}}$.
First we observe $f(x_k) \to f(x)$, since $f(x_k)$ is not increasing and has $f(x)$ as an accumulation point due to the continuity of f. For all $N \in \mathbb{N}$ we calculate

$$\begin{aligned} f(x) - f(x_0) \ &\leq \ f(x_{N+1}) - f(x_0) \leq \sum_{k=0}^{N} f(x_{k+1}) - f(x_k) \\ &\leq \sum_{k=0}^{N} -\delta \sigma_k \|D_k\| \leq -\delta \sum_{k=0}^{N} \varepsilon_k h(\varepsilon_k), \end{aligned}$$

so $h(\varepsilon_k)\varepsilon_k \to 0$ as $k \to 0$ and this means

$$\varepsilon_k \to 0 \ , \tag{2.30}$$

since h is nondecreasing.
Now we assume $0 \notin \partial f(x)$. Again as pointed out in Lemma 1.64 we can find ε, $K > 0$ such that for all $f' \in \partial^{\varepsilon} f(x)$ we have $\|f'\| > K$. We define $r := \frac{\varepsilon}{2}$.
Then for every $i, k \in \mathbb{N}$ the assumptions $\varepsilon_{k,i} < r$ and $x_k \in B_X(x, r)$ imply $\|D_{k,i}\| > K$, since $D_{k,i} \in \partial^{\varepsilon_{k,i}} f(x_k) \subseteq \partial^{\varepsilon} f(x)$.

   (i) First we assume that there exists a k_0 such that for all $k \geq k_0$ hold $x_k \in B_X(x, r)$

and lead this assumption to a contradiction.
If $x_k \in B_X(x, r)$ we have that for fixed k the term $\varepsilon_{k,i}$ becomes only decreased with respect to i if

$$\varepsilon_{k,i} \geq \min\left\{\inf\left\{\varepsilon \;\middle|\; h(\varepsilon) \geq K\right\}, r\right\} =: \varepsilon^0 > 0,$$

because only in this case it is possible that (NSAb) holds, since otherwise

$$\|D_{k,i}\| > K > h(\varepsilon_{k,i}).$$

So we gain with $\varepsilon_{k+1,i} \geq H(\varepsilon_{k+1,i-1})$ for $i > 0$ and the monotonicity of H that[14]

$$\varepsilon_{k+1} = \varepsilon_{k+1,i} \geq \min\left\{\varepsilon_{k+1,0}, H(\varepsilon_{k+1,i-1})\right\} \geq \min\left\{g(\|D_k\|, \varepsilon_k), H(\varepsilon^0)\right\}.$$

Since $\varepsilon_{k+1} \to 0$ and $H(\varepsilon^0) > 0$ there exists some k_1 such that for all $k > k_1$ holds $H(\varepsilon^0) > \varepsilon_{k+1}$ and therefore

$$\varepsilon_{k+1} \geq g(\|D_k\|_k, \varepsilon_k).$$

This gives a contradiction to Assumption 2.24 since:

(a) $K < \|D_k\| \to 0$ as $k \to \infty$ or

(b) there exists some $k_2 > k_1$ such that for all $k > k_2$ holds $\varepsilon_{k+1} \geq \varepsilon_k$ and so $\varepsilon_k \nrightarrow 0$ or

(c) there exists $n_0 > k_1$ such that $\varepsilon_n < r$ holds additionally to $x_n \in B_X(x, r)$ for all $n > n_0$. Therefore

$$\begin{aligned} f(x) - f(x_{n_0}) &\leq f(x_{n+1}) - f(x_{n_0}) \leq \sum_{k=n_0}^{n} f(x_{k+1}) - f(x_k) \\ &\leq \sum_{k=n_0}^{n} -\delta\sigma_k \|D_k\| \\ &\leq -\delta K \sum_{k=n_0}^{n} \varepsilon_k \\ &\leq -\delta K \sum_{k=n_0}^{n} g_K^{(k-n_0)}(\varepsilon_{n_0}) \to -\infty, \end{aligned}$$

[14] We distinguish the cases $i = 0$ and $i > 0$, so we just set $\varepsilon_{k+1,-1} := \varepsilon^0$.

Which is a contradiction too. (The above inequality

$$f(x) - f(x_{n_0}) \leq -\delta K \sum_{k=n_0}^{n} \varepsilon_k$$

also shows that if we choose $\varepsilon_{k,0}$ as in Remark 2.27, we also get a contradiction using again $\varepsilon_k > \min\left\{\varepsilon_{k,0},\, H(\varepsilon^0)\right\}$.)

(ii) Now we assume the sequence leaves $B_X(x\,,\,r)$ infinitely often. Since x is an accumulation point of $(x_k)_{k\in\mathbb{N}}$ and $x_k \nrightarrow x$ we find some R with $r > R > 0$ such that for all $k \in \mathbb{N}$ there exists some $k' > k$ such that $x_{k'} \notin B_X(x\,,\,R)$. We choose a subsequence $(x_{k(i)})_{i\in\mathbb{N}}$ such that

$$x_{k(2j)} \in B_X\left(x\,,\,\frac{R}{2}\right),\; x_{k(2j+1)} \notin B_X(x\,,\,R)\,,\; x_l \in B_X(x\,,\,R) \subseteq B_X(x\,,\,r)$$

for $k(2j) \leq l < k(2j+1)$ and $\varepsilon_l < R < r$ for all $l > k(0)$. We obtain with the abbreviation $I_N := \left\{j \in \mathbb{N} \,\middle|\, k(2j+1) < N\right\}$ that

$$\begin{aligned}
f(x) - f(x_0) &= \lim_{N\to\infty} f(x_N) - f(x_0) \\
&\leq \lim_{N\to\infty} \sum_{j\in I_N} f(x_{k(2j+1)}) - f(x_{k(2j)}) \\
&\leq \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} f(x_{l+1}) - f(x_l) \\
&\leq \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} -\delta \left\|D_l\right\| \left\|x_{l+1} - x_l\right\| \\
&\leq -\delta K \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} \left\|x_{l+1} - x_l\right\| \\
&\leq -\delta K \lim_{N\to\infty} \sum_{j\in I_N} \left\|x_{k(2j+1)} - x_{k(2j)}\right\| \\
&\leq -\delta K \lim_{N\to\infty} \sum_{j\in I_N} \frac{R}{2} = -\infty,
\end{aligned}$$

which is a contradiction.

Since we get a contradiction in both cases (i) and (ii), we obtain $0 \in \partial f(x)$.

$\diamondsuit$

In the literature one often requires of other algorithms that the level set $\{x \in X \mid f(x) \leq f(x_0)\}$ is compact, e.g. f is coercive and X is finite dimensional, which gives us the existence of accumulation points. And if we require that there is exactly one critical point x satisfying $f(x) = \inf\{f(x_k) \mid k \in \mathbb{N}\}$ in these level sets, the existence of an accumulation point gives the convergence of the algorithm to this critical point in a finite dimensional space X. We formulate now results similar to Satz 4.4.11 in W. Alt's introduction,[1].

Proposition 2.31 (accumulation points)
Let X be a Banach space, $(x_k)_{k\in\mathbb{N}} \in X^{\mathbb{N}}$ a sequence and AP the set of accumulation points of $(x_k)_{k\in\mathbb{N}}$. If $\{x_k \mid k \in \mathbb{N}\}$ is relatively compact, then $AP \neq \emptyset$ and

$$dist(x_k, AP) := \inf\{\|y - x_k\| \mid y \in AP\} \to 0 \text{ as } k \to 0.$$

If in addition $\|x_k - x_{k+1}\| \to 0$ as $k \to \infty$, either $AP = \{x\}$, i.e. $(x_k)_{k\in\mathbb{N}}$ converges to x, or AP has no isolated points.

Proof. Assume $dist(x_k, AP) \nrightarrow 0$, then there exists some $K > 0$ and a subsequence $(x_{k(i)})_{i\in\mathbb{N}}$ with $dist(x_{k(i)}, AP) > K$ $(i \in \mathbb{N})$. Due to the relative compactness of our sequence, we can choose a further subsequence of our subsequence which is convergent to some $x \in AP$, which gives a contradiction to the definition of $dist(\cdot, AP)$.
Now we assume $\|x_k - x_{k+1}\| \to 0$. If AP has an isolated point x, then we can find some $r > 0$ such that $B_X(x, r) \cap AP = \{x\}$. By above, for some $k_0 \in \mathbb{N}$ we have $x_{k_0} \in B(x, \frac{r}{4})$, since x is an accumulation point, and for all $k > k_0$ we have $dist(x_k, AP) < \frac{r}{4}$ and $\|x_k - x_{k+1}\| < \frac{r}{4}$. By induction we see that for all $l \geq k$ it holds $x_l \in B_X\left(x, \frac{r}{4}\right)$, which follows since by construction we have $dist(x_k, AP) \leq \frac{r}{4}$ and since $\|x_k - x_{k+1}\| < \frac{r}{4}$ implies $x_{k+1} \in B(x, \frac{r}{2})$ and therefore for all $x \neq y \in AP$ we have $\|y - x_{k+1}\| > \frac{r}{2}$. But this means that x is the only accumulation point.
♢

As a direct consequence we obtain:

Proposition 2.32
Let X be a reflexive Banach space, $f : X \to \mathbb{R}$ be locally Lipschitz continuous and $(x_k)_{k\in\mathbb{N}}$ a sequence in X produced by Algorithm 2.26 such that $\{x_k \mid k \in \mathbb{N}\}$ is relatively compact and $\|x_k - x_{k+1}\| \to 0$ as $k \to \infty$.
Then either $(x_k)_{k\in\mathbb{N}}$ is convergent to a critical point or the set of accumulation points contains only critical points and none isolated points.
If $\{x \in X \mid f(x) \leq f(x_0)\}$ contains only finitely many critical points then $(x_k)_{k\in\mathbb{N}}$ converges to a critical point in $\{x \in X \mid f(x) \leq f(x_0)\}$.

PROOF. Since X is reflexive, by Theorem 2.29, every accumulation point is a critical point. So if there are only finitely many, which means that every accumulation point is isolated, we obtain that $(x_k)_{k\in\mathbb{N}}$ converges to a critical point in $\left\{x \in X \;\middle|\; f(x) \leq f(x_0)\right\}$. ♢

Remark 2.33
- One can easily ensure that $\|x_k - x_{k+1}\| \to 0$ by requiring in every descent step additionally to (DSAb2) that e.g. $\sigma_k = \|x_k - x_{k+1}\| \leq \alpha(\varepsilon_k)$ for some $\alpha : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ with $\lim_{t\to 0} \alpha(t) = 0$, since the proof of Theorem 2.29 showed that $\varepsilon_k \to 0$. But in practice this is usually not necessary.
- In the case $X = \mathbb{R}^n$ and $\left\{x \in X \;\middle|\; f(x) \leq f(x_0)\right\}$ is bounded it holds $\left\{x_k \;\middle|\; k \in \mathbb{N}\right\}$ is relatively compact and X is reflexive.

2.3 Specialized Descent Algorithm Based on Approximating Models

In applications one is often in the situation that the energy function is of the form

$$f = f_1 + f_0,$$

where f_1 is a C^1 function and f_0 is Lipschitz continuous, but not differentiable, or has high curvature. E.g. we could add to a smooth function a nonsmooth penalty function to deal with constraints or in contact mechanics we could consider friction. In those examples one is often in the situation that computing the gradient of f_1 and its Hessian is computationally expensive, but it can be approximated well by a quadratic function locally. In this case one would like to use Trust-Region methods to approximate f_1 by a model m, which has a simple form on the Trust-Region radius, e.g. we typically approximate f_1 by a linear or quadratic functional. Later, in order to compute $D_{k,i}$, we have to compute several elements of the $\varepsilon-$gradients of f at the point x_k. With the approximation m of f_1 it becomes computationally much easier to compute approximations of those elements instead of the elements itself. In algorithms for contact mechanics one has often to compute the gradient and Hessian of f_1 at every point x_k anyway. Therefore a natural approximation is given and computing a gradient of f_1, which is typically solving a linear equation, becomes a simple matrix vector multiplication. In the following we will discuss a way to ensure convergence to a critical point, if we approximate f_1 in every iteration by a model function. We will not approximate f_0, because we could not develop a useful theory for this purpose. Thus the difficulties computing gradients of f_0 remain. In some applications, like e.g. some friction models, computing the gradient of f_0 is not that expansive since friction only occurs at the boundary. This is not the case in our later computations, but in several applications this seems to be the case.
To be more precise, we give a definition of a model, which we will use in this work.

Definition 2.34 Let X be a Banach space and $f : X \to \mathbb{R}$ be a locally Lipschitz continuous function. We call a locally Lipschitz continuous function $m : X \to \mathbb{R}$ a **suitable approximating model** of f in $x \in X$, if the gradient is sufficiently good approximated

$$\limsup_{\varepsilon \to 0} \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon}(f-m)(x) \right\} = 0$$

and the model approximates f itself locally sufficiently good in the sense that for all $\vartheta > 0$ we have

$$\lim_{\varepsilon \to 0} \inf_{y \in B(x,\varepsilon) \cap M_{<\vartheta}(x)} \frac{f(y) - f(x)}{m(y) - m(x)} \geq 1, \tag{locA}$$

where define $\inf \emptyset := \infty$ and $M_{<\vartheta(x)}$, to avoid technical complications regarding dividing by zero:

$$M_{<\vartheta}(x) := \left\{ y \in X \;\middle|\; m(y) - m(x) < -\vartheta \left\| x - y \right\| \right\}.$$

In the case $0 \in \partial m(x)$ it is possibe that $M_{<\vartheta(x)} = \emptyset$ for all $\vartheta > 0$. In this case is (locA) always satisfied.

Remark 2.35

- The notation m for the model function is standard in the literature, c.f. [15].
- We do not demand that $f(x) = m(x)$, but of course one can always add a constant to m.
- Proposition 1.7 implies $\partial(f-m)(x) = \{0\}$ and so Proposition 1.10 implies $\partial f(x) = \partial m(x)$ since we can apply the inclusion in both directions.
- We motivate the set $M_{<\vartheta}(x)$ in the case $0 \notin \partial m(x)$. By Lemma 1.64 for arbitraty $\varepsilon > 0$ there exists some $0 < \tilde{\varepsilon} < \varepsilon$, some $K > 0$ and some $d \in S_X(0,1)$ with $\left\langle f' \,\middle|\, d \right\rangle > K$ for every $f' \in \partial^{\tilde{\varepsilon}} m(x)$. Now Lebourg's Theorem implies that for every $\vartheta < K$ and every $0 < t < \tilde{\varepsilon}$ we have $x - td \in M_{<\vartheta}(x)$. This means that in the case $0 \notin \partial m(x)$ we have $B(x,\varepsilon) \cap M_{<\vartheta}(x) \neq \emptyset$ for sufficiently small ϑ.

We give one example.

Example 2.36 (example for a suitable model)
If $f = f_1 + f_0$, f_1 is a C^1 function and f_0 is Lipschitz continuous, then

$$m(x+y) := \left\langle f_1'(x) \,\middle|\, y \right\rangle + \left\langle Ay \,\middle|\, y \right\rangle + f_0(x+y)$$

is an approximating model of f in $x \in X$, where $A : X \to X'$ is any continuous linear mapping, usually an approximation of the Hessian of f_1 at the point x. But we can also choose $A = 0$, if we do not want to (or can not) calculate such an A.

PROOF. Since $f-m \in C^1$ we calculate (knowing that a continuous convex function on a convex set attains its maximum at extreme points):

$$\begin{aligned}
& \sup\left\{\left\|f'\right\| \;\middle|\; f' \in \partial^{\varepsilon}(f-m)\,(y)\right\} \\
= \; & \sup\left\{\left\|f'\right\| \;\middle|\; f' \in \bigcup_{y\in\overline{B(x\,,\,\varepsilon)}} \partial(f-m)\,(y)\right\} \\
= \; & \sup_{y\in\overline{B(x\,,\,\varepsilon)}} \left\|f_1'(y)-f_1'(x)-(A+A^T)(y-x)\right\| \\
\to \; & 0
\end{aligned}$$

as $\varepsilon \to 0$. If $y \in M_{<\vartheta}(x)$ we find with the Mean Value Theorem and $y = x + (y-x)$ some $\xi_y \in]x,y[$ such that

$$\begin{aligned}
& \left|1-\frac{f(y)-f(x)}{m(y)-m(x)}\right| \\
= \; & \left|1-\frac{\left\langle f_1'(\xi_y)\,\middle|\,y-x\right\rangle + f_0(y)-f_0(x)}{m(y)-m(x)}\right| \\
= \; & \left|1-\frac{\left\langle f_1'(\xi_y)-f_1'(x)-A(y-x)\,\middle|\,y-x\right\rangle + m(y)-m(x)}{m(y)-m(x)}\right| \\
= \; & \left|\frac{\left\langle f_1'(\xi_y)-f_1'(x)-A(y-x)\,\middle|\,y-x\right\rangle}{m(y)-m(x)}\right| \\
\leq \; & \frac{1}{\vartheta}\left\|f_1'(\xi_y)-f_1'(x)-A(y-x)\right\| \to 0
\end{aligned}$$

as $\|y-x\| \to 0$, which gives the second assumption. ♢

Let us now specialize our basic descent Algorithm 2.26. We call it specialisation because every sum of a smooth and a nonsmooth function is nonsmooth again. Of course one can also see it the other way around. With $f_1 = 0$ we regain our basic descent Algorithm 2.26, except for the fact that we now also allow the norm to change in every iteration.

Remark 2.37 In the following algorithm we will consider in very iteration step k a possibily different norm $\|\cdot\|_k$. For $f' \in X'$ we use the notation

$$\|f'\|_k := \sup\left\{\left\langle f' \,\middle|\, x\right\rangle \;\middle|\; x \in X \;:\; \|x\|_k = 1\right\}.$$

The reason why we consider a change of the norms is to apply our theory to the Newton method. We recall that the Newton method is a Steepest Descent method where we change the norm

in every step. Therefore we also generalize the Newton method (or the Semismooth Newton method of M. Ulbrich, [56, 57]) with our ansatz. For further details we refer to the later Chapter 4. There we also need to take a new $\varepsilon_{k,0}$ such that $\varepsilon_{k,0} \geq g(\|f'\|_k, \varepsilon_{k-1})$ for some $f' \in \partial f(x_k)$ in the later Step 2.

Algorithm 2.38 (specialized descent algorithm)

1. *Initialization: Choose h, H and g satisfying Assumption 2.24,*

$$\delta, \gamma \in]0,1[\,,\ x_0 \in X, \varepsilon_{0,0} =: \varepsilon_{-1} > 0$$

   and set $i = k = 0$.

2. *Choose a suitable approximating model $m = m_k$ of f in x_k.*

3. *Choose a norm $\|\cdot\|_k$ on X which is equivalent to $\|\cdot\|$, such that $(X, \|\cdot\|_k)$ and its dual space are strictly convex and if one likes, one can choose a new $\varepsilon_{k,0} \geq g(\|f'\|_k, \varepsilon_{k-1})$ for some $f' \in \partial f(x_k)$.*

4. *Choose $D_{k,i} \in \partial^{\varepsilon_{k,i}} m(x_k)$ and a dual $d_{k,i} \in X$ of $D_{k,i}$ with $\|d_{k,i}\| = 1$ such that either the descent step assumption for m*

$$m(x_k - \varepsilon_{k,i} d_{k,i}) - m(x_k) \leq -\delta \|D_{k,i}\|_k \varepsilon_{k,i} \tag{DSAs}$$

   or the null step assumption

$$\|D_{k,i}\|_k < h(\varepsilon_{k,i}) \tag{NSAs}$$

   is satisfied.

5. *In the case that*

$$\|D_{k,i}\|_k < h(\varepsilon_{k,i})$$

   or the model is an insufficiently good approximation of f in the sense that

$$\gamma\,(m(x_k - \varepsilon_{k,i} d_{k,i}) - m(x_k)) < f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) \tag{iMA}$$

   choose $\varepsilon_{k,i+1}$ sufficiently smaller than $\varepsilon_{k,i}$, i.e. $\varepsilon_{k,i} \to 0$ if $i \to \infty$ for fixed k, but not too small in the sense that

$$\varepsilon_{k,i+1} \geq H(\varepsilon_{k,i})\ ;$$

   e.g. $\varepsilon_{k,i+1} := H(\varepsilon_{k,i})$ satisfies both conditions; increment i by one and go to 4.

6. *Set* $x_{k+1} = x_k - \sigma_k d_k$ *for some* $\sigma_k \geq \varepsilon_k := \varepsilon_{k,i}$ *such that the two inequalities*

$$\begin{aligned} f(x_k - \sigma_k d_{k,i}) - f(x_k) &\leq \gamma \left(m(x_k - \sigma_k d_{k,i}) - m(x_k) \right) \\ &\leq -\gamma\delta \left\| D_{k,i} \right\|_k \sigma_k \end{aligned} \tag{DSAs2}$$

hold and choose

$$\varepsilon_{k+1,0} \geq g(\left\| D_k \right\|_k, \varepsilon_k)$$

where $D_k := D_{k,i}$*, set* $i = 0$*, increment* k *by one and go to 2.*

Remark 2.39 Instead of choosing $\varepsilon_{k+1,0} \geq g(\left\| D_k \right\|_k, \varepsilon_k)$ one could also choose $\varepsilon_{k+1,0}$ such that $\sum\limits_{k \in \mathbb{N}} \varepsilon_{k,0} = \infty$.

Remark 2.40 In the later Theorem 2.44 we will see that such $D_{k,i}$ and $d_{k,i}$ exist and that for fixed $k \in \mathbb{N}$ (NSAs) and (iMA) can both hold only finitely many times, so we make only finite many null steps.

We need a further assumption on the choice of the approximation functions, to gain convergence. We need that the properties of a suitable model function are uniform in k.

Assumption 2.41 (necessary assumptions for spec. algorithm)

- *The norms* $\left\| \cdot \right\|_k$ *and* $\left\| \cdot \right\|$ *are uniformly equivalent, thus there exists some* $K_{eq} \geq 1$ *such that for all* $k', k \in \mathbb{N}$ *we have*

$$\left\| \cdot \right\|_k \leq K_{eq} \left\| \cdot \right\|_{k'} \leq K_{eq}^2 \left\| \cdot \right\| \leq K_{eq}^3 \left\| \cdot \right\|_k .$$

- *Assume that for every accumulation point* $x \in X$ *of a sequence* $(x_k)_{k \in \mathbb{N}}$ *created by Algorithm 2.38 and every* $K > 0$*,* $C < 1$ *and every* $\vartheta > 0$ *there exists some* $\varepsilon > 0$ *such that for all* $k \in \mathbb{N}$ *with* $x_k \in B(x, \varepsilon)$ *hold*

$$\sup \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon}(m_k - f)(x_k) \right\} < K$$

and

$$\inf_{y \in B(x,\varepsilon) \cap M^k_{<\vartheta}(x_k)} \frac{f(y) - f(x_k)}{m_k(y) - m_k(x_k)} \geq C,$$

where we define again

$$M^k_{<\vartheta}(x_k) := \left\{ y \in X \;\middle|\; m_k(y) - m_k(x_k) < -\vartheta \left\| x_k - y \right\|_k \right\}.$$

Example 2.42 (example for a model)
If $f = f_1 + f_0$, f_1 is a C^1 function and f_0 is Lipschitz continuous, and $A_k : X \to X'$ are linear continuous mappings then

$$m_k(x_k + y) := \langle f_1'(x_k) \,|\, y \rangle + \langle A_k y \,|\, y \rangle + f_0(x_k + y)$$

is an approximating model of f which satisfies the second part of Assumption 2.41, if $\|A_k\|$ is uniformly bounded in k.

Proof. For fixed $K > 0$, $\vartheta > 0$, $C < 1$ choose $\varepsilon > 0$ such that for all $y, z \in \overline{B_X(x\,,\,2\varepsilon)}$ holds

$$\left\| f_1'(y) - f_1'(z) \right\| + \sup_{k \in \mathbb{N}} \|A_k\| \, \varepsilon \leq \frac{1}{K_{eq}} \min\{K,\, \vartheta(1 - C)\} \ ,$$

then the same computations as in the proof that this model is suitable in every point give the claim. ♢

Example 2.43 *A trivial example is of course $m_k = f$.*

We now show that every accumulation point of the sequence produced by the algorithm is a critical point.

Theorem 2.44 (accumulation points are critical points)
Let X be a reflexive Banach space, $f : X \to \mathbb{R}$ be a locally Lipschitz continuous function and x_k iteration points produced by Algorithm 2.38.

1. *For fixed $k, i \in \mathbb{N}$ there exist $D_{k,i} \in \partial^{\varepsilon_{k,i}} m_k(x_k)$ and a dual $d_{k,i}$ satisfying (DSAs) or (NSAs).*
2. *If $0 \notin \partial f(x_k)$ for some fixed $k \in \mathbb{N}$, then there exists only finitely many $i \in \mathbb{N}$ such that (NSAs) or (iMA) is satisfied.*
3. *For any sequence $(x_k)_{k \in \mathbb{N}} \in X^{\mathbb{N}}$ produced by Algorithm 2.38, which satisfies Assumption 2.41, the sequence $(f(x_k))_{k \in \mathbb{N}}$ is strictly decreasing and for every accumulation point x of $(x_k)_{k \in \mathbb{N}}$ we have $0 \in \partial f(x)$ and $f(x_k) \to f(x)$ as $k \to \infty$.*

Proof. The proof is similar to the proof of Theorem 2.29. Although the ideas are the similar, we have to write it down again and place emphasis on the new aspects, since new technical

aspects have to be considered.

1. The proof of the first statement is literally the same as in Theorem 2.29 with f replaced by m_k. In the case $0 \in \partial m_k(x_k)$ we simply choose $D_{k,i} = 0$ which satisfies (NSAs) and in the case $0 \notin \partial m_k(x_k)$ we can apply Lebourg's Theorem to find a suitable $D_{k,i}$ to satisfiy (DSAs).

2. Now we assume $0 \notin \partial f(x_k) = \partial m_k(x_k)$ for fixed $k \in \mathbb{N}$. By Lemma 1.64 we can find ε, $K > 0$ such that for all $m' \in \partial^\varepsilon m_k(x_k)$ we have $\|m'\| > K$. Since $\varepsilon_{k,i} \to 0$ as $i \to \infty$ in the case that we make infinitely many null steps and thus $h(\varepsilon_{k,i}) \to 0$, we obtain that

$$K \leq \|D_{k,i}\| \leq h(\varepsilon_{k,i}) \ ,$$

thus (NSAs), can only hold for finitely many $i \in \mathbb{N}$ as above. Therefore we obtain that (DSAs) holds almost ever by construction, thus

$$m_k(x_k - \varepsilon_{k,i} d_{k,i}) - m_k(x_k) \leq -\delta \|D_{k,i}\| \varepsilon_{k,i} < -\frac{1}{2}\delta K \varepsilon_{k,i} \ ,$$

which means by definition that $x_k - \varepsilon_{k,i} d_{k,i} \in M_{<\frac{\delta}{2}K}(x_k)$ almost ever. Since m_k is suitable the assumption (locA) tells us in the case that we would make infinitely many null steps we would have

$$1 \leq \liminf_{i\to\infty} \frac{f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k)}{m_k(x_k - \varepsilon_{k,i} d_{k,i}) - m_k(x_k)} \ .$$

We have choosen $0 < \gamma < 1$ in the initialization of the algorithm, thus

$$\gamma \left(m_k(x_k - \varepsilon_{k,i} d_{k,i}) - m_k(x_k)\right) \ < \ f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) \ ,$$

i.e. (iMA), can hold only finitely many times too.

3. $(f(x_k))_{k\in\mathbb{N}}$ is strictly decreasing by construction. Assume x is an accumulation point of $(x_k)_{k\in\mathbb{N}}$.
First we observe $f(x_k) \to f(x)$, since $f(x_k)$ is not increasing and has $f(x)$ as an accumulation point due to the continuity of f. For all $N \in \mathbb{N}$ we calculate

$$f(x) - f(x_0) \ \leq \ f(x_{N+1}) - f(x_0) \leq \sum_{k=0}^{N} f(x_{k+1}) - f(x_k)$$

$$\leq \sum_{k=0}^{N} -\gamma\delta\sigma_k \|D_k\|_k \leq -\gamma\delta \sum_{k=0}^{N} \varepsilon_k h(\varepsilon_k),$$

so $h(\varepsilon_k)\varepsilon_k \to 0$ as $k \to 0$ and this means $\varepsilon_k \to 0$, since h is nondecreasing.

Assume $0 \notin \partial f(x)$. As pointed out in Lemma 1.64 we can find ε, $K > 0$ such that for all $f' \in \partial^{2\varepsilon} f(x)$ holds $\|f'\| > 2K$.

By Assumption 2.41 we can choose ε such smaller that on the one hand further for all $k \in \mathbb{N}$ with $x_k \in B_X(x, \varepsilon)$ and for all $(m-f)' \in \partial^{\varepsilon}(m_k - f)(x_k)$ holds $\|(f-m)'\| < K$ and on the other hand for our concrete $\gamma \in]0,1[$ we have

$$\inf_{y \in B(x,\varepsilon) \cap M^k_{<(\delta K)}(x)} \frac{f(y) - f(x_k)}{m_k(y) - m_k(x_k)} \geq \gamma. \tag{2.45}$$

Moreover we observe as direct consequence of the definition of $\partial^{\varepsilon} f$ and Proposition 1.10 that for all $y \in B_X(x, \varepsilon)$ holds

$$\partial^{\varepsilon} m_k(y) \subseteq \partial^{\varepsilon} f(y) + \partial^{\varepsilon}(m_k - f)(y) \subseteq \partial^{2\varepsilon} f(x) + \partial^{\varepsilon}(m_k - f)(y).$$

This inequalities imply that for all $x_k \in B_X(x, \varepsilon)$ and all $m' \in \partial^{\varepsilon} m_k(x_k)$ exists some $f' \in \partial^{\varepsilon} f(x_k)$ and some $(m-f)' \in \partial^{\varepsilon}(m_k - f)(x_k)$ and we have

$$\|m'\| = \|f' + (m-f)'\| \geq \|f'\| - \|(m-f)'\| > 2K - K = K.$$

Then for every $i, k \in \mathbb{N}$ the assumptions $\varepsilon_{k,i} < \varepsilon$ and $x_k \in B_X(x, \varepsilon)$ imply

$$D_{k,i} \in \partial^{\varepsilon_{k,i}} m_k(x_k) \subseteq \partial^{\varepsilon} m(x_k)$$

and therefore

$$\|D_{k,i}\|_k > \frac{K}{K_{eq}} := K'. \tag{2.46}$$

(i) First we assume that there exists a k_0 such that for all $k \geq k_0$ hold $x_k \in B_X(x, \varepsilon)$ and lead this assumption to a contradiction.
We will show next that in the case $x_k \in B_X(x, \varepsilon)$ we have that

$$\varepsilon_{k,i} \geq \min\left\{\inf\left\{\varepsilon \,\middle|\, h(\varepsilon) \geq K'\right\}, \varepsilon\right\} =: \varepsilon^0$$

is necessary for both of the inequalities (NSAs) and (iMA). Therefore $\varepsilon_{k,i}$ becomes

only decreased for fixed k with respect to i in the case $\varepsilon_{k,i} \geq \varepsilon^0 > 0$.
So let us assume $\varepsilon_{k,i} < \varepsilon^0$. Then $\varepsilon_{k,i} \leq \inf\left\{\varepsilon \,\middle|\, h(\varepsilon) \geq K'\right\}$ implies

$$\|D_{k,i}\|_k > K' > h(\varepsilon_{k,i})$$

by (2.46). This means (NSAs) is not satisfied and so $D_{k,i}$ would have been chosen in such a way that:

$$m_k(x_k - \varepsilon_{k,i} d_{k,i}) - m_k(x_k) \leq -\delta \|D_{k,i}\| \varepsilon_{k,i} < -\delta K \varepsilon_{k,i},$$

which implies with $\varepsilon_{k,i} < \varepsilon$ by definition

$$x_k - \varepsilon_{k,i} d_{k,i} \in M^k_{<(\gamma K)}(x_k) \cap B_X(x_k, \varepsilon) \ ,$$

and so by (2.45)

$$\gamma\left(m_k(x_k - \varepsilon_{k,i} d_{k,i}) - m_k(x_k)\right) \geq f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) \ .$$

This means (iMA) is not true too, which gives the claim and that we decrease $\varepsilon_{k+1,i}$ only in the case $\varepsilon_{k,i} \geq \varepsilon^0 > 0$.

So for some $i \in \mathbb{N}$ we gain with[15] $\varepsilon_{k+1,i} \geq H(\varepsilon_{k+1,i-1})$ for $i > 0$ and the monotonicity of H [16]

$$\varepsilon_{k+1} = \varepsilon_{k+1,i} \geq \min\left\{\varepsilon_{k+1,0}, H(\varepsilon_{k+1,i-1})\right\} \geq \min\left\{g(\|D_k\|_k, \varepsilon_k), H(\varepsilon^0)\right\}.$$

Since $\varepsilon_{k+1} \to 0$ and $H(\varepsilon^0) > 0$ there exists some k_1 such that for all $k > k_1$ holds $H(\varepsilon^0) > \varepsilon_{k+1}$ and therefore

$$\varepsilon_{k+1} \geq g(\|D_k\|_k, \varepsilon_k) \ .$$

This gives a contradiction to Assumption 2.24 since:

(a) $K < \|D_k\|_k \to 0$ as $k \to \infty$ or

(b) there exists some $k_2 > k_1$ such that for all $k > k_2$ holds $\varepsilon_{k+1} \geq \varepsilon_k$ and so $\varepsilon_k \nrightarrow 0$ or

[15] We distinguish the cases $i = 0$ and $i > 0$, so we just set $\varepsilon_{k+1,-1} := \varepsilon^0$.

[16] If we choose some new $\varepsilon_{k+1,0} \geq g(\|f'_{k+1}\|_{k+1}, \varepsilon_k)$ for some $f'_{k+1} \in \partial f(x_{k+1})$ in Step 3, then the assumption implies $\|f'_{k+1}\|_{k+1} \to 0$, which gives $0 \in \partial f(x)$.

(c) there exists $n_0 > k_1$ such that $\varepsilon_n < r$ holds additionally to $x_n \in B_X(x, r)$ for all $n > n_0$. Therefore

$$\begin{aligned} f(x) - f(x_{n_0}) &\leq f(x_{n+1}) - f(x_{n_0}) \leq \sum_{k=n_0}^{n} f(x_{k+1}) - f(x_k) \\ &\leq \sum_{k=n_0}^{n} -\gamma\delta\sigma_k \|D_k\|_k \leq^{(2.46)} -\gamma\delta K' \sum_{k=n_0}^{n} \varepsilon_k \\ &\leq -\gamma\delta K' \sum_{k=n_0}^{n} g_{K'}^{k-n_0}(\varepsilon_{n_0}) \to -\infty \,. \end{aligned}$$

Which is a contradiction too. (The above inequality

$$f(x) - f(x_{n_0}) \leq -\gamma\delta K' \sum_{k=n_0}^{N} \varepsilon_k$$

shows further that in the situation of Remark 2.39 we also get a contradiction using $\varepsilon_k > \min\left\{\varepsilon_{k,0},\, H(\varepsilon^0)\right\}$.)

(ii) Now we consider the case that the sequence does not remain finally in the ball. Since x is an accumulation point of $(x_k)_{k\in\mathbb{N}}$ and $x_k \nrightarrow x$ we find some R with $\varepsilon > R > 0$ such that for all $k \in \mathbb{N}$ there exists some $k' > k$ such that $x_{k'} \notin B_X(x, R)$. We choose a subsequence $(x_{k(i)})_{i\in\mathbb{N}}$ such that

$$x_{k(2j)} \in B_X\left(x, \frac{R}{2}\right),\ x_{k(2j+1)} \notin B_X(x, R),\ x_l \in B_X(x, R) \subseteq B_X(x, \varepsilon)$$

for $k(2j) \leq l < k(2j+1)$ and $\varepsilon_l < R < \varepsilon$ for all $l > k(0)$. We obtain with the abbreviation $I_N := \left\{j \in \mathbb{N} \,\middle|\, k(2j+1) < N\right\}$ that

$$\begin{aligned} f(x) - f(x_0) &= \lim_{N\to\infty} f(x_N) - f(x_0) \\ &\leq \lim_{N\to\infty} \sum_{j\in I_N} f(x_{k(2j+1)}) - f(x_{k(2j)}) \\ &\leq \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} f(x_{l+1}) - f(x_l) \\ &\leq \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} -\gamma\delta \|D_l\|_k \|x_{l+1} - x_l\|_k \end{aligned}$$

$$
\begin{aligned}
&\leq -\gamma\delta K' \lim_{N\to\infty} \sum_{j\in I_N} \sum_{l=k(2j)}^{k(2j+1)-1} \|x_{l+1} - x_l\|_k \\
&\leq -\gamma\delta K' \lim_{N\to\infty} \sum_{j\in I_N} \left\|x_{k(2j+1)} - x_{k(2j)}\right\|_k \\
&\leq -\gamma\delta K' \lim_{N\to\infty} \sum_{j\in I_N} K_{eq}^{-1} \frac{R}{2} = -\infty.
\end{aligned}
$$

Since we get again a contradiction in both case (i) and (ii), we obtain $0 \in \partial f(x)$.

♢

Of course we can also generalize Proposition 2.32.

Proposition 2.47
Let X be a reflexive Banach space, $f : X \to \mathbb{R}$ be locally Lipschitz continuous and $(x_k)_{k\in\mathbb{N}}$ a sequence in X produced by Algorithm 2.38 such that $\{x_k \mid k \in \mathbb{N}\}$ is relatively compact and $\|x_k - x_{k+1}\| \to 0$ as $k \to \infty$.
Then either $(x_k)_{k\in\mathbb{N}}$ is convergent to a critical point or the set of accumulation points contains only critical points and none isolated points.
If $\{x \in X \mid f(x) \leq f(x_0)\}$ contains only finitely many critical points then $(x_k)_{k\in\mathbb{N}}$ converges to a critical point in $\{x \in X \mid f(x) \leq f(x_0)\}$.

Proof. The proof is analog to the proof of Proposition 2.32. ♢

3 Inner Approximation of $\partial^\varepsilon f(x)$

Now we give a procedure to calculate the elements $D_{k,i}$ of the generalized gradients of the neighborhood of x_k, such that one of the equations (DSAb) or (NSAb) in our basic Algorithm 2.26 (or one of the equations (DSAs) or (NSAs) in our specialized Algorithm 2.38) is satisfied.

3.1 The Hilbert Space Case

First we motivate the algorithm in the Hilbert space case. Corollary 1.59 tells us that it is sufficient to find some $f' \in \partial^\varepsilon f(x_k)$ with sufficiently small norm. For this purpose let us make the following observation.

Lemma 3.1 *Let H be a Hilbert space and choose $\delta \in [0,1[$, $L > 0$. Further let $(a_k)_{k\in\mathbb{N}}$ and $(b_k)_{k\in\mathbb{N}}$ be sequences in H with the following properties:*

1. $\langle a_k \,|\, b_k \rangle \leq \delta \|a_k\|^2 \quad (k \in \mathbb{N})$,
2. $\|b_k\| \leq L \quad (k \in \mathbb{N})$,
3. $\|a_{k+1}\| \leq \min\limits_{\lambda\in[0,1]} \|\lambda a_k + (1-\lambda) b_k\|$.

Then $\|a_k\| \to 0$ as $k \to \infty$.

Proof. By construction, the mapping $k \mapsto \|a_k\|$ is non increasing. W.l.o.g. $\dim H > 1$, elsewise we have $\|a_{k+1}\| \leq \delta \|a_k\|$ and therefore $(a_k)_{k\in\mathbb{N}}$ is a null sequence. Further we assume w.l.o.g. that $L > \|a_0\|$.
We prove now:

$$\|a_{k+1}\|^2 \leq \frac{\|a_k\|^2 L^2}{(1-\delta)^2 \|a_k\|^2 + L^2} \text{ for all } k \in \mathbb{N}\,. \tag{3.2}$$

For this, we decompose H in $\mathrm{Lin}_{\mathbb{R}}\{a_k\}$ and its orthogonal complement H_k. This means, we can write $b_k = s_k a_k + t_k \tilde{b}_k$ where $s_k \leq \delta$, $L \geq t_k \geq 0$ and $\tilde{b}_k \in H_k$ with $\left\|\tilde{b}_k\right\| = 1$. With the abbreviation $r_k := \|a_k\|$ we have for all $\lambda \in [0,1]$

$$\|\lambda a_k + (1-\lambda) b_k\|^2 \leq (\lambda + (1-\lambda)s_k)^2 r_k^2 + (1-\lambda)^2 L^2 =: \alpha_k(\lambda)\,.$$

α_k is convex and therefore λ_0 is a minimizer of α_k iff $\alpha_k'(\lambda_0) = 0$, hence in the case:

$$\begin{aligned} & 2(\lambda_0(1-s_k)+s_k)(1-s_k)r_k^2 + 2(\lambda_0 - 1)L^2 = 0 \\ \Leftrightarrow\quad & \lambda_0 \left[(1-s_k)^2 r_k^2 + L^2\right] = (-s_k)(1-s_k)r_k^2 + L^2, \end{aligned}$$

$$\Leftrightarrow \quad \lambda_0 = 1 - \frac{(1-s_k)r_k^2}{(1-s_k)^2 r_k^2 + L^2},$$

thus $\lambda_0 < 1$, since $s_k \leq \delta < 1$. Further $\lambda_0 > 0$, since

$$(-s_k)(1-s_k)r_k^2 + L^2 > -r_k^2 + L^2 \geq -\|a_0\|^2 + L^2 \ .$$

We calculate further

$$\begin{aligned}
\|a_{k+1}\|^2 &\leq \alpha_k(\lambda_0) \\
&= (\lambda_0(1-s_k) + s_k)^2 r_k^2 + (1-\lambda_0)^2 L^2 \\
&= \left(\frac{(-s_k)(1-s_k)^2 r_k^2 + (1-s_k)L^2}{(1-s_k)^2 r_k^2 + L^2} + s_k\right)^2 r_k^2 + \frac{L^2(1-s_k)^2 r_k^4}{((1-s_k)^2 r_k^2 + L^2)^2} \\
&= \frac{L^4 r_k^2}{((1-s_k)^2 r_k^2 + L^2)^2} + \frac{L^2(1-s_k)^2 r_k^4}{((1-s_k)^2 r_k^2 + L^2)^2} \\
&= \frac{L^2 r_k^2}{(1-s_k)^2 r_k^2 + L^2} \\
&\leq \frac{L^2 r_k^2}{(1-\delta)^2 r_k^2 + L^2} \quad \text{since } s_k \leq \delta.
\end{aligned}$$

Thus we have shown the inequality (3.2). Since $(\|a_k\|)_{k\in\mathbb{N}}$ is a positive non increasing sequence, the sequence converges towards some A. The inequality (3.2) tells us that

$$A^2 \leq \frac{A^2 L^2}{(1-\delta)^2 A^2 + L^2} \Leftrightarrow (1-\delta)^2 A^4 + A^2 L^2 \leq A^2 L^2 \Rightarrow A = 0 \ .$$

Therefore $(a_n)_{n\in\mathbb{N}}$ converges to 0. ♢

Now let us look at the Assumption (DSAb), i.e.

$$f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) \leq -\delta \|D_{k,i}\| \varepsilon_{k,i}$$

for some $D_{k,i} \in \partial^{\varepsilon_{k,i}} f(x_k)$ and its dual element $d_{k,i}$. Since for the moment we consider Hilbert spaces this means $d_{k,i} = D_{k,i} \frac{1}{\|D_{k,i}\|}$. If f is differentiable in x_k, linearization would now suggest that $D \in \partial f(x_k)$ is a good first candidate for $D_{k,i}$, at least when $\varepsilon_{k,i}$ is small. If f is not differentiable, any $D \in \partial f(x_k)$ would be a reasonable first guess for $D_{k,i}$. The idea is now to create a reasonable sequence $(a_k)_{k\in\mathbb{N}}$ of good guesses, with $a_0 \in \partial f(x_k)$ and $a_j \in \partial^{\varepsilon_{k,i}} f(x_k)$. If (DSAb) is satisfied with the choice $D_{k,i} = a_0$, we have found $D_{k,i}$ and we are done. So let us assume that the inequality is not satisfied. With Lebourg's Theorem it follows that if (DSAb)

is not satisfied, we can find some $\xi \in [x_k, x_k - \frac{\varepsilon_{k,i}}{\|a_0\|} a_0]$ and some $b_0 \in \partial f(\xi)$ such that

$$-\varepsilon_{k,i} \left\langle b_0 \,\middle|\, d_{k,i} \right\rangle = f(x_k - \varepsilon_{k,i} d_{k,i}) - f(x_k) > -\delta \|D_{k,i}\| \varepsilon_{k,i}$$

and so the assumption

$$\left\langle a_0 \,\middle|\, b_0 \right\rangle \leq \delta \|a_0\|^2$$

of Lemma 3.1 is satisfied. Since a_0 was not good enough, we try

$$a_1 = \arg\min \left\{ \|a\| \;\middle|\; a \in [a_0, b_0] \right\}$$

as second guess. Again, if (DSAb) is satisfied with the choice $D_{k,i} = a_1$, we are done. If not, by Lebourgs Theorem we can find some $\xi \in [x_k, x_k - \frac{\varepsilon_{k,i}}{\|a_1\|} a_1]$ and some $b_1 \in \partial f(\xi)$ such that

$$\left\langle a_1 \,\middle|\, b_1 \right\rangle \leq \delta \|a_1\|^2$$

holds. Now we choose $a_2 = \arg\min \left\{ \|a\| \;\middle|\; a \in [a_1, b_1] \right\}$ and start again. This leads to the following algorithm.

Algorithm 3.3 (inner approximation for Hilbert spaces)
For fixed $k, i \in \mathbb{N}$ and fixed $x_k \in X$, $\varepsilon_{k,i} > 0$, $1 > \delta > 0$ and fixed $h(\varepsilon_{k,i}) > 0$ do:

1. *Choose any $a_0 \in \partial f(x_k)$ and any $\delta' \in]\delta, 1[$ and set $j = 0$.*

2. *If the inequality*
$$f\left(x_k - \frac{\varepsilon_{k,i}}{\|a_j\|} a_j\right) - f(x_k) \leq -\delta \|a_j\| \varepsilon_{k,i}$$
is satisfied or
$$\|a_j\| \leq h(\varepsilon_{k,i}) \,,$$
set $D_{k,i} = a_j$ and stop.

3. *Determine with the following Algorithm 3.9 some $\xi \in [x_k, x_k - \frac{\varepsilon_{k,i}}{\|a_j\|} a_j]$ and some $b_j \in \partial f(\xi)$ such that*
$$\left\langle a_j \,\middle|\, b_j \right\rangle \leq \delta' \|a_j\|^2 \,.$$

4. *Choose some $C_j \subseteq \{a_l \,|\, l \leq j\} \cup \{b_l \,|\, l \leq j\}$ with $a_j, b_j \in C_j$ and define the* **inner approximation**
$$A_j := conv\, C_j = \overline{conv}^*\, C_j \subseteq \partial^{\varepsilon_{k,i}} f\,(x_k) \,.$$

5. *Compute*

$$a_{j+1} := \arg\min \left\{ \left\| f' \right\| \;\middle|\; f' \in A_j \right\} \ ,$$

increment j by one and go to 2.

We usually use this algorithm in our applications to compute the descent direction of our basic descent Algorithm 2.26 or its specialization Algorithm 2.38. As mentioned above, we will generalize this algorithm to Sobolev spaces and its closed subspaces later. But it turns out that in applications it is much better to calculate in a Hilbert space setting, at least from the numerical point of view, because in a Hilbert space we can determine the dual element easily and exactly. Now we state that the algorithm terminates.

Proposition 3.4 *In the case that f is Lipschitz on some neighborhood of $B_X(x_k, \varepsilon_{k,i})$ and Algorithm 3.9 always terminates, Algorithm 3.3 stops after finitely many steps.*

Proof. This follows directly from Lemma 3.1, where L denotes the Lipschitz constant, and that the algorithm stops as soon as $\|a_j\| \leq h(\varepsilon_{k,i})$. ♢

Remark 3.5

- A_j is automatically a subset of $\partial^{\varepsilon_{k,i}} f(x_k)$ since $\partial^{\varepsilon_{k,i}} f(x_k)$ is by definition convex and $b_l \in \partial f(\xi_j) \subseteq \partial^{\varepsilon_{k,i}} f(x_k)$ for $l \in \mathbb{N}$. This implies that we always have

$$\|a_j\| \geq \min \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon_{k,i}} f(x_k) \right\} \ .$$

- We show now that $\|a_j\| \to \min \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon_{k,i}} f(x_k) \right\}$ as $j \to \infty$ would imply that $a_j \to \arg\min \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon_{k,i}} f(x_k) \right\}$ as $j \to \infty$. With other words, the a_j are somehow approximations of the optimal descent direction which is the element with the smallest norm. We recall that all gradients in a neighborhood of the optimal descent direction are also descent directions on the entire set $B_X(x_k, \varepsilon_{k,i})$, cf. Corollary 1.59.
  Since we consider Hilbert spaces (or later strictly convex, reflexive Banach spaces with strictly convex dual space)

$$a := \arg\min \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon_{k,i}} f(x_k) \right\}$$

  exists and is unique, since $\partial^{\varepsilon_{k,i}} f(x_k)$ is convex and closed. Assume we do not have

$$a_j \to \arg\min \left\{ \left\| f' \right\| \;\middle|\; f' \in \partial^{\varepsilon_{k,i}} f(x_k) \right\} \ .$$

Due to the boundedness of $(a_j)_{j\in\mathbb{N}}$ there exists a weakly convergent subsequence with $\|a_{j(l)} - a\| > r$ for some $r > 0$. Weak convergence and convergence of the norm imply strong convergence on Hilbert spaces, thus we have a contradiction. (In the later situation, where the Banach spaces and the dual space are uniformly convex, we can apply [12, Chapter II, Prop. 2.8].)

- There are various minimization algorithms to determine a_{j+1} in Step 5. As examples we recall the projected gradient methods and the semismooth Newton type method in combination with the Fischer-Burmeister functions, cf. Ulbrich [57, 56]. Also the sequential quadratic programming (SQP) methods can be used to solve this problem, cf. [1, 44]. To apply these algorithms, we enumerate the elements of C_j, i.e. $C_j = \{c_1, c_2, \ldots, c_m\}$ for some $m \in \mathbb{N}$. Then we use these methods to solve the quadratic minimization problem in $\lambda := (\lambda_i)_{i\le m} \in \mathbb{R}^m$

$$\left\| \sum_{i=1}^{m} \lambda_i c_i \right\|^2 =: (\lambda_i)_{i\le m}^T \cdot A \cdot (\lambda_i)_{i\le m} \to \min \tag{3.6}$$

under the constraints

$$\sum_{i=1}^{m} \lambda_i = 1 \text{ and } \lambda_i \ge 0 \tag{3.7}$$

for all $i \le m$, where $A = (a_{k,l})_{k,l\le m}$ with $a_{k,l} := \langle c_k \,|\, c_l \rangle$. Then every solution $(\lambda_i^0)_{i\le m}$ of this minimization problem gives

$$a_{j+1} = \sum_{i=1}^{m} \lambda_i^0 c_i \ . \tag{3.8}$$

This minimization problem is an easy problem, which creates no difficulties in solving such that solving this problem is very cheap regarding computational time, at least for small m.

- In practice we usually take

$$C_j := \{a_0\} \cup \{a_l \,|\, j - m \le l \le j\} \cup \{b_j\}$$

or

$$C_j \subseteq \{a_j,\ a_0\} \cup \{b_l \,|\, j - m \le l \le j\} \ ,$$

where $m \approx 10$. The reason for that is that while working on the later benchmark problems

and the 1−Laplacian problem, we got the best results regarding convergence for this choice. But we have to mention that we did not investigate the question of the optimal choice for C_j systematically and we leave this task for upcoming work. We can not analytically explain, why it is useful to keep a_0, but it appears to make a substantial difference. Taking a large set C_j (typically $m > 30$) leads to a numerical inaccuracy of a_{j+1} up to our experience, which we also can not explain analytically either.(This problem is also known from the Gradient Sampling algorithm.) Moreover in this case we seem to need a larger number of gradients. We further recommend that the choice of m depends on the dimension of X. We can of course also take $C_j = \{a_j,\ b_j\}$. In this case it is an easy task to determine a_{j+1} analytically, but we are wasting a lot of information which we have already gained.

3.2 An Algorithm to find b_j

Next we give an algorithm, to compute the b_j in Algorithm 3.3, i.e. the algorithm has not stopped and so for some a_j we have

$$f\left(x_k - \frac{\varepsilon_{k,i}}{\|a_j\|}a_j\right) - f(x_k) > -\delta \|a_j\| \varepsilon_{k,i}$$

and we seek for some $\xi \in \left[x_k - \frac{\varepsilon_{k,i}}{\|a_j\|}a_j, x_k\right]$ and some $b_j \in \partial f(\xi)$ such that

$$\left\langle a_j \,\middle|\, b_j \right\rangle \leq \delta' \|a_j\|^2 \ .$$

We formulate the algorithm generally in a Banach space, since we need it later again for the Sobolov spaces. Also to simplify notation, we ignore the notation of Algorithm 3.3 for a moment.

Algorithm 3.9 *Let X be a Banach space, $f : X \to \mathbb{R}$ be locally Lipschitz continuous and $x, d \in X$, $0 < \delta < \delta' < 1$, $K > 0$ and*

$$f(x - d) - f(x) > -\delta K \|d\| \ . \tag{3.10}$$

Our goal is to determine some $\xi \in [x, x - d]$ such that

$$\left\langle f' \,\middle|\, d \right\rangle \leq \delta' K \|d\|$$

for some $f' \in \partial f(\xi)$.

1. *Choose any $f_1' \in \partial f(x - d)$ and set $l = 1$ and $R_1 := x - d$ and $L_1 := x$.*
2. *If $\left\langle f_1' \,\middle|\, d \right\rangle \leq \delta' K \|d\|$ stop and return f_1'.*

3. *Choose any* $f_l' \in \partial f\left(\frac{R_l+L_l}{2}\right)$.

4. *If*

$$\left\langle f_l' \,\middle|\, d \right\rangle \leq \delta' K \|d\|$$

stop and return f_l'.

5. *If*

$$f\left(\frac{L_l+R_l}{2}\right) - f(L_l) > -\delta K \left\| \frac{L_l - R_l}{2} \right\| \tag{3.11}$$

then set $R_{l+1} = \frac{L_l+R_l}{2}$ *and* $L_{l+1} = L_l$.
Otherwise it holds

$$f(R_l) - f\left(\frac{L_l+R_l}{2}\right) > -\delta K \left\| \frac{L_l - R_l}{2} \right\| \tag{3.12}$$

(cf. proof of Proposition 3.17) and we set $L_{l+1} = \frac{L_l+R_l}{2}$ *and* $R_{l+1} = R_l$.
Notice that in both cases

$$f\left(R_{l+1}\right) - f(L_{l+1}) > -\delta K \left\| L_{l+1} - R_{l+1} \right\| . \tag{3.13}$$

(cf. (3.10)).
Alternatively to (3.11) we could test

$$f\left(\frac{L_l+R_l}{2}\right) - f(L_l) \geq f(R_l) - f\left(\frac{L_l+R_l}{2}\right) \ , \tag{3.14}$$

which is even stronger than (3.11) (cf. proof of Proposition 3.17).

6. *Increment* l *by* 1 *and go to 3.*

Thus with the special choices

$$x = x_k,\ d := \frac{\varepsilon_{k,i}}{\|a_j\|} a_j,\ \|d\| = \varepsilon_{k,i},\ K := \|a_j\|$$

in Algorithm 3.3 we are in the situation of Algorithm 3.9 and for the returned $b_j := f_l'$ the inequality $\left\langle f_l' \,\middle|\, d \right\rangle \leq \delta' K \|d\|$ becomes the demanded inequality $\left\langle b_j \,\middle|\, a_j \right\rangle \leq \delta' \|a_j\|^2$ of Algorithm 3.3.

Remark 3.15 1. We point out that we do not have to determine a concrete representation of $f_l' \in X'$ with respect to some given basis of X' in every iteration step. This would be by far too expensive. In practice we choose only formally some $f_l' \in \partial f\left(\frac{L_l+R_l}{2}\right)$ and only compute the term $f_l'(d) = \left\langle f_l' \,\middle|\, d \right\rangle$ which we normally know analytically. Only in the case

that the stopping criterion is satisfied we calculate a concrete representation of f_l' with respect to the given basis of X'.
If f is differentiable, then $\tilde{f}_l : \mathbb{R} \to \mathbb{R}$ with

$$\tilde{f}_l(t) := f(y_l - td) \ , \ \ y_l := \frac{L_l + R_l}{2}$$

is differentiable and

$$\tilde{f}_l'(0) = \left\langle f'(y_l) \,\middle|\, -d \right\rangle \ .$$

Thus, for Step 1, 2, 3 and 4, we usually only need to compute a simple scalar derivative.
In the general case of a locally Lipschitz continuous function we know that

$$\partial \tilde{f}_l(0) \subseteq \left\langle \partial f(y_l) \,\middle|\, -d \right\rangle \ .$$

Moreover $\tilde{f}_l$ is differentiable almost everywhere on a neighborhood of $t = 0$. In numerical computations one could tacitly assume that $\tilde{f}_l{}'(0)$ exists and would compute a discretized approximation of it. This way we would implicitly select some $\tilde{f}_l' \in \partial \tilde{f}_l(0)$ and some $f_l' \in \partial f(y_l')$ in Step 3. But often we know more about the function and can exploit more structure.
E.g. in applications we work on the Sobolev space $X = H^1(\Omega)$ for some open and bounded Ω and consider e.g. f having the form

$$f(x) := \int_\Omega G(x) d\omega$$

with some $G : \mathbb{R} \to \mathbb{R}$. Under suitable smoothness assumptions we have

$$\tilde{f}_l'(0) = \left\langle f'(y_l) \,\middle|\, -d \right\rangle = -\int_\Omega G'(y_l) \cdot d \, d\omega \ .$$

Since G' is usually known, this is a simple computation.
In our later applications we also consider $G(x) := |x|$, which is regular and satisfies Hypothesis A before [13, Theorem 2.7.5]. So [13, Theorem 2.7.5] gives

$$f_l' := \left(h \mapsto \int_\Omega \operatorname{sgn}(y_l) \cdot h \, d\omega \right) \in \partial f(y_l)$$

for every measurable $\operatorname{sgn} : \mathbb{R} \to \mathbb{R}$ given by $\operatorname{sgn}(0) \in [-1, 1]$ and $\operatorname{sgn}(x) := \frac{x}{|x|}$ else. Computing $f_l'(d)$ is a simple computation and sufficient for Algorithm 3.9.

But in order to compute the norm of f_l' or the dual element of f_l' we have to compute the Riesz representation $R(f_l')$ of f_l'. (For a Hilbert space H we understand under the Riesz representation the image of the Riesz mapping $R : H' \to H$, which allows us to identify H' and H, cf. [59].) In the case $G(x) := |x|$ this means we have to solve the weak equation: For all $h \in H^1(\Omega)$

$$\left\langle R(f_l') \,\middle|\, h \right\rangle_{H^1(\Omega)} = \int_\Omega \operatorname{sgn}(y_l) \cdot h \, d\omega \; .$$

Later, when we consider finite dimensional subspaces of $H^1(\Omega)$, computing the Riesz representation with respect to the finite element basis is by far the most time consuming computation. Therefore we are glad, that we only have to compute $R(f_l')$ for the last f_l' in Algorithm 3.9. $R(f_l')$ is the return value of Algorithm 3.9 in our computational praxis, since only with this representation we can solve actually the minimization problem in Step 5 of Algorithm 3.3 efficiently.
For the benchmark problems in Chapter 5 the set $\partial f(y_l)$ and the Riesz representation of single element $f_l' \in \partial f(y_l)$ are easily computed via an explicit formula, since we consider on $\mathbb{R}^n$ the Euclidean norm. So there the situation is much simpler.

2. Let us now motivate that Algorithm 3.9 terminates in practice. In the general cases of a locally Lipschitz continuous function f we know that $\tilde{f} : \mathbb{R} \to \mathbb{R}$ with $\tilde{f}(t) := f(x - td)$ is Lipschitz continuous on $[0, 1]$ and, thus, absolute continuous on $[0, 1]$ and differentiable almost everywhere on $[0, 1]$. By the generalized Fundamental Theorem of Calculus (c.f. [51, Satz 7.20]) and by (3.10)

$$\int_0^1 \tilde{f}'(t)dt = f(x - d) - f(x) > -\delta K \|d\| \; .$$

Hence there is a subset $J \subseteq [0, 1]$ with positive measure such that for all $t \in J$

$$\tilde{f}'(t) > -\delta K \|d\| \; .$$

If $L_l = x - t_l d$, then $R_l = x - \tilde{t}_l d$ with $\tilde{t}_l - t_l = \frac{\|R_l - L_l\|}{\|d\|}$ and

$$f(R_l) - f(L_l) = \int_{t_l}^{\tilde{t}_l} \tilde{f}'(t)dt \; .$$

By (3.13)

$$\int_{t_l}^{\tilde{t}_l} \tilde{f}'(t)dt > -\delta K \|R_l - L_l\|$$

and thus

$$\tilde{f}'(t) > -\delta K \|d\| \tag{3.16}$$

on a subset $J_l \subseteq I_l := [t_l, \tilde{t}_l]$ having positive measure. This means that $I_l \cap J$ always has positive measure and, generically, we would expect that $\hat{t}_l := \frac{t_l + \tilde{t}_l}{2} \in J$ after finitely many steps.
Since $\tilde{f}'(\hat{t}_l) \in \langle \partial f(x - \hat{t}d) \mid -d \rangle$, there is $f_l' \in \partial f(x - \hat{t}_l d)$ such that

$$\tilde{f}'(\hat{t}_l) = \langle f_l' \mid -d \rangle \overset{(3.16)}{>} -\delta K \|d\| \ .$$

So we would generically expect the algorithm to stop after Step 4.
In practice one often has that f is continuously differentiable on an open and dense set D, which implies that f is strictly differentiable on this set. But if f is strictly differentiable at this $\hat{t}_l$, the element f_l' is unique, cf. Proposition 1.8, so we have in Step 4 of the algorithm

$$\langle f_l' \mid d \rangle = -\tilde{f}'\left(\hat{t}_l\right) \overset{(3.16)}{<} \delta K \|d\| < \delta' K \|d\|$$

and the algorithm stops.
Unfortunately we can not expect that Algorithm 3.9 always determines, since we are dealing with arbitrary Lipschitz functions. One easily gives a rather artificial example where the algorithm does not terminate even so $f : \mathbb{R} \to \mathbb{R}$ is differentiable in the accumulation point and continuously differentiable on $\mathbb{R} \setminus \{0\}$, cf. Remark 3.20.

But we have the following.

Proposition 3.17 *Let X be a Banach space, $f : X \to \mathbb{R}$ be locally Lipschitz continuous and $x, d \in X$, $0 < \delta < \delta' < 1$, $K > 0$ such that (3.10) holds.*

1. *If Algorithm 3.9 does not terminate and therefore produces sequences $(L_l)_{l\in\mathbb{N}}$ and $(R_l)_{l\in\mathbb{N}}$ which converge to some $\xi_\infty \in [x, x - d]$, then there exists some $f' \in \partial f(\xi_\infty)$ with*

$$\langle f' \mid d \rangle \leq \delta' K \|d\|$$

and f is not strictly differentiable at ξ_∞.

2. *If f is convex on a neighborhood of $[x, x-d]$, then Algorithm 3.9 already terminates in Step 2.*

PROOF.

1. We prove first by induction that for all $l \geq 1$ holds

$$f(R_l) - f(L_l) > -\delta K \|L_l - R_l\| \ . \tag{3.18}$$

For $l = 1$ this holds by assumption (3.10). Now assume (3.18) holds for $l \geq 1$. We have to consider four cases: (3.11) and not (3.11) in the original version and (3.14) and not (3.14) in the second formulation with the alternative choice in Step 5.

(a) If (3.11) in the first version holds, we choose $R_{l+1} = \frac{L_l+R_l}{2}$ and $L_{l+1} = L_l$ and inserting into (3.11) gives (3.18) for $l+1$.

(b) (3.14) and (3.18) imply

$$\begin{aligned}
& 2\left(f\left(\frac{L_l+R_l}{2}\right) - f(L_l)\right) \\
= & \left(f\left(\frac{L_l+R_l}{2}\right) - f(L_l)\right) + \left(f\left(\frac{L_l+R_l}{2}\right) - f(L_l)\right) \\
\geq & \left(f\left(\frac{L_l+R_l}{2}\right) - f(L_l)\right) + \left(f(R_l) - f\left(\frac{L_l+R_l}{2}\right)\right) \\
= & f(R_l) - f(L_l) \\
> & -\delta K \|L_l - R_l\| \ ,
\end{aligned}$$

thus (3.11) holds again. In the alternative formulation we choose $R_{l+1} = \frac{L_l+R_l}{2}$ and $L_{l+1} = L_l$ in the case (3.14) too. So then we obtain (3.18) for $l+1$ too.

(c) An analog estimate gives that not (3.14) implies together with (3.18) that

$$2\left(f(R_l) - f\left(\frac{L_l+R_l}{2}\right)\right) > f(R_l) - f(L_l) > -\delta K \|L_l - R_l\| \ ,$$

thus (3.12). In the alternative formulation we choose $L_{l+1} = \frac{L_l+R_l}{2}$ and $R_{l+1} = R_l$ in the case not (3.14). Thus then we obtain the claimed (3.18) for $l+1$.

(d) (3.18) and not (3.11) imply not (3.14) by contra position applied to the case (1b). Therefore (1c) gives (3.12). Since now $L_{l+1} = \frac{L_l+R_l}{2}$ and $R_{l+1} = R_l$ too we obtian the claimed (3.18) for $l+1$.

Therefore induction gives the claim.
By the Theorem of Lebourg there exist some $\xi_l \in [L_l, R_l]$ and $\tilde{f}'_l \in \partial f(\xi_l)$ with

$$\begin{aligned} -\delta K \|L_l - R_l\| &< f(R_l) - f(L_l) \\ &= \left\langle \tilde{f}'_l \,\middle|\, R_l - L_l \right\rangle \\ &= \left\langle \tilde{f}'_l \,\middle|\, -\frac{\|R_l - L_l\|}{\|d\|} d \right\rangle . \end{aligned}$$

By $(f'_l)_{l\in\mathbb{N}}$ we denote the sequence produced by the algorithm. Since the algorithm does not stop and since $d \neq 0$ due to (3.10) we obtain:

$$\left\langle \tilde{f}'_l \,\middle|\, d \right\rangle < \delta K \|d\| < \delta' K \|d\| < \left\langle f'_l \,\middle|\, d \right\rangle . \tag{3.19}$$

$L_l \to \xi_\infty$ and $R_l \to \xi_\infty$ give that also $\xi_l \to \xi_\infty$. Since $(f'_l)_{l\in\mathbb{N}}$ and $(\tilde{f}'_l)_{l\in\mathbb{N}}$ are bounded by the Lipschitz continuity near ξ_∞, some subsequences converge weakly to some f' and some $\tilde{f}'$ respectively, which are both in $\partial f(\xi_\infty)$ by the upper semicontinuity of the generalized gradient, cf. Proposition 1.7. If f would be strictly differentiable at ξ_∞ we would have that $\partial f(\xi_\infty)$ has only one element, cf. Proposition 1.8. Thus $f' = \tilde{f}'$, what gives a contradiction to (3.19) after taking the limit.

2. We recall that for convex and Lipschitz continuous functions the subdifferential and Clarke's gradient coincide, cf. Proposition 1.8. So the statement for convex functions follows with the observation that for every $f' \in \partial f(x - d)$ we have by the definition of the subdifferential, cf. [13],

$$\delta' K \|d\| > \delta K \|d\| > f(x) - f(x - d) \geq \left\langle f' \,\middle|\, d \right\rangle .$$

♢

Remark 3.20 One can easily construct such a situation, where Algorithm 3.9 does not terminate after finitely many steps, as e.g. the simple example

$$f(t) = -t^2 \sin\left(\frac{2\pi}{t}\right)$$

with $f(0) := 0$ and $K = d = 1$ and $x = 0$ shows.
(Induction directly gives that till the algorithm stops, it is

- $R_l = -2^{1-l}$,

- $L_l = 0$
- $f(L_l) = f(R_l) = f\left(\frac{R_l+L_l}{2}\right) = 0$ and
- (3.11) holds,

since then $R_{l+1} = \frac{R_l+L_l}{2} = -2^{-l}$ and $L_{l+1} = 0$. Further we have

$$f'(t) = -2t \sin\left(\frac{2\pi}{t}\right) + 2\pi \cos\left(\frac{2\pi}{t}\right)$$

for $t \neq 0$, thus $f_l' = 2\pi$ and $\left\langle f_l' \,\middle|\, d \right\rangle = 2\pi > \delta = \delta K \|d\|$. Therefore the algorithm doesn't stop.) However in what we consider as practical situation this effect does not occur.

Remark 3.21 We recall that at the computer we work with finite dimensional Banach spaces X. Typically f is continuously differentiable on an open and dense set D, thus the choice of f_l' in Step 3 of Algorithm 3.9 is normally unique. If this is not the case, i.e. f_l' is not given uniquely in Step 3 of Algorithm 3.9 then we are free to choose any generalized gradient. There is no reason to compute the norm smallest generalized gradient. First of all it would often be computationally expensive. And second we want primarily that a_{j+1} in Algorithm 3.3 becomes as small as possible with respect to the norm, since we want to approximate the optimal descent direction, which is the norm smallest element of the gradient at the neighbourhood. But taking $b_j := f_l'$ in Algorithm 3.3 smallest with respect to the norm, does not imply that a_{j+1} does become so. We omit here giving an explicit example, although one could easily give some.

3.3 The Banach Space Case

Next we wish to formulate Algorithm 3.3 for Banach spaces. To do so, we need a generalization of Lemma 3.1 for Banach spaces. This turned out to be too ambitious for us for arbitrary Banach spaces. So we tried to find a suitable class of Banach spaces to prove a generalization of Lemma 3.1. It seems a natural idea to formulate a generalization for uniformly convex Banach spaces. But then we need in our proofs strong assumptions on the modulus of convexity, which are not even met for the Hilbert space. But for subspaces of the Sobolev spaces with $1 < p < \infty$, we can formulate a generalization. This we will do next, since we want to find the first eigenfunctions of the p-Laplacian as minimal points of a function defined on the Sobolev space in the case $p > 1$. Having the theory for a Sobolev space at hand it does not matter whether we first discretize and then optimize or the other way arround. So we can leave the controversially debated choice of Ansatz to the user. When we later compute an approximation of the first eigenfunctions of the 1-Laplacian we will not have this choice since we solve a

minimization problem on the space of bounded variation. The norm for the space of bounded variation is not strictly convex, thus we can not apply most of our results to this space. Therefore we solve the resulting minimization problem by first discretizing the space of bounded variation. The resulting finite dimensional minimization problem we solve with an equivalent norm having the properties we require. Typically we use an Hilbert space norm, because then computing the gradient becomes easier and fast as we will see later, cf. Section 6. But we could also use the norm of a Sobolev space as in [29, 28] due to the following Lemma.

Lemma 3.22 *Let $(X, \|\cdot\|)$ be a closed subspace of the Sobolev space $W^{k,p}(\Omega)$, with $p \in]1, \infty[$, $k,\ n \in \mathbb{N}$ and Ω be an open subset of $\mathbb{R}^n$. Here we use the norm*

$$\|u\|^p := \sum_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_p}^p$$

or in the case that it is a norm

$$\|u\|^p := \sum_{|\alpha| = k} \|\partial^\alpha u\|_{L_p}^p$$

where $\|u\|_{L_p}^p := \int\limits_\Omega |u|^p(x)dx$. Let $(a'_k)_{k\in\mathbb{N}}$ and $(b'_k)_{k\in\mathbb{N}}$ be two sequences in $X' \setminus \{0\}$. We denote by $(j(a'_k))_{k\in\mathbb{N}}$ the dual sequence of $(a'_k)_{k\in\mathbb{N}}$, i.e.

$$j(a'_k) := \arg\max \left\{ \left\langle a'_k \,\middle|\, d \right\rangle \;\middle|\; d \in \overline{B_X(0,1)} \right\}$$

and $\left\langle a'_k \,\middle|\, j(a'_k) \right\rangle = \|a'_k\|$.
If there exist constants $\gamma \in [0, 1[$ and $L > 0$ such that for all $k \in \mathbb{N}$

1. $\left\langle b'_k \,\middle|\, j(a'_k) \right\rangle \leq \gamma \|a'_k\|$,
2. $\|b'_k\| \leq L$,
3. $\left\|a'_{k+1}\right\| \leq \min\limits_{\lambda\in[0,1]} \|\lambda a'_k + (1-\lambda) b'_k\|$,

then $\|a'_k\| \to 0$ as $k \to \infty$.

Remark 3.23

- The lemma is of course also true for $\mathbb{R}^n$ with the $p-$norm, the l_p and the L_p since they can be seen as subspaces of $L_p(\Omega) =: W^{0,p}(\Omega)$.
- Further we recall that it does not make sense to look at the case $p = 1.0$ or $p = \infty$ here since in this case the norm is not strictly convex, which we need for our basic Algorithm 2.26.

In the proof of Lemma 3.22 we need the Clarkson inequality. For $1 < p < \infty$ we define $p' := \frac{p}{p-1}$, i.e. $\frac{1}{p} + \frac{1}{p'} = 1$. Then for every $u, v \in L_p(\Omega)$ the Clarkson inequality states in the case $1 < p \leq 2$

$$2\left(\|x\|^{p'} + \|y\|^{p'}\right)^{p-1} \leq \|x+y\|^p + \|x-y\|^p \leq 2(\|x\|^p + \|y\|^p) \tag{3.24}$$

and in the case $p \geq 2$

$$2\left(\|x\|^{p'} + \|y\|^{p'}\right)^{p-1} \geq \|x+y\|^p + \|x-y\|^p \geq 2(\|x\|^p + \|y\|^p)\,, \tag{3.25}$$

cf. [14]. It seems to be know that the Clarkson inequatities hold also for Sobolev spaces, but we could not find a rigid prove. Further is appears to be known that if the Clarkson inequalities hold for some Banach space X for some $1 < p < \infty$, then the Clarkson inequalities hold also on the dual space X' for the above p', cf. [31, Theorem 2.9] or Lemma 3.27. Next we state those results and prove them. The reason why we state and prove those results here is that we couldn't find entirely correct proofs in the literature, even though many people consider them as proven. After that we will prove Lemma 3.22.

Lemma 3.26 *The Clarkson inequalities hold for Sobolov spaces $W^{k,p}(\Omega)$ with $1 < p < \infty$ and the norm*

$$\|u\|^p := \sum_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_p}^p$$

and for Sobolov spaces $W_0^{1,p}(\Omega)$ with $1 < p < \infty$ and the norm

$$\|u\|^p := \sum_{|\alpha| = 1} \|\partial^\alpha u\|_{L_p}^p \ .$$

Proof. The Clarkson inequalities (3.24) and (3.25) hold on $L_p(\Omega)$ for $1 < p < \infty$ and $\frac{1}{p} + \frac{1}{p'} := 1$, c.f. [14]. We only prove Clarkson for the norm $\|u\|^p := \sum\limits_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_p}^p$, since the proof for the other norm is analog.

- We consider the case $1 < p \leq 2$ first. We estimate with the triangle inequality for $\mathbb{R}^2$ with the $\frac{p'}{p}$-norm:

$$2\left(\|u\|^{p'} + \|v\|^{p'}\right)^{p-1} \quad = \quad 2\left(\left(\sum_{|\alpha| \leq k} \|\partial^\alpha u\|_{L_p}^p\right)^{\frac{p'}{p}} + \left(\sum_{|\alpha| \leq k} \|\partial^\alpha v\|_{L_p}^p\right)^{\frac{p'}{p}}\right)^{\frac{p}{p'}}$$

$$
\begin{aligned}
&= 2\left\| \begin{pmatrix} \sum_{|\alpha|\le k} \|\partial^\alpha u\|_{L_p}^p \\ \sum_{|\alpha|\le k} \|\partial^\alpha v\|_{L_p}^p \end{pmatrix} \right\|_{\frac{p'}{p}} \\
&= 2\left\| \sum_{|\alpha|\le k} \begin{pmatrix} \|\partial^\alpha u\|_{L_p}^p \\ \|\partial^\alpha v\|_{L_p}^p \end{pmatrix} \right\|_{\frac{p'}{p}} \\
&\le 2\sum_{|\alpha|\le k} \left\| \begin{pmatrix} \|\partial^\alpha u\|_{L_p}^p \\ \|\partial^\alpha v\|_{L_p}^p \end{pmatrix} \right\|_{\frac{p'}{p}} \\
&= 2\sum_{|\alpha|\le k} \left(\|\partial^\alpha u\|_{L_p}^{p'} + \|\partial^\alpha v\|_{L_p}^{p'} \right)^{\frac{p}{p'}} \\
&\overset{(3.24)}{\le} \sum_{|\alpha|\le k} \|\partial^\alpha (u+v)\|_{L_p}^p + \|\partial^\alpha (u-v)\|_{L_p}^p \\
&= \|u+v\|^p + \|u-v\|^p \ .
\end{aligned}
$$

Further we calculate

$$
\begin{aligned}
\|u+v\|^p + \|u-v\|^p &= \sum_{|\alpha|\le k} \left(\|\partial^\alpha (u+v)\|_{L_p}^p + \|\partial^\alpha (u-v)\|_{L_p}^p \right) \\
&\overset{(3.24)}{\le} \sum_{|\alpha|\le k} 2(\|\partial^\alpha u\|_{L_p}^p + \|\partial^\alpha v\|_{L_p}^p) \\
&= 2(\|u\|^p + \|v\|^p) \ .
\end{aligned}
$$

- Now we consider the case $p \ge 2$. By N we denote the cardinality of $\left\{\alpha \,\middle|\, |\alpha| \le k\right\}$. With this we calculate

$$
\begin{aligned}
&\|u+v\|^p + \|u-v\|^p = \\
&= \sum_{|\alpha|\le k} \left(\|\partial^\alpha (u+v)\|_{L_p}^p + \|\partial^\alpha (u-v)\|_{L_p}^p \right) \\
&\overset{(3.25)}{\le} \sum_{|\alpha|\le k} 2\left(\|\partial^\alpha u\|_{L_p}^{p'} + \|\partial^\alpha v\|_{L_p}^{p'} \right)^{p-1} \\
&= 2\left(\sum_{|\alpha|\le k} \left(\|\partial^\alpha u\|_{L_p}^{p'} + \|\partial^\alpha v\|_{L_p}^{p'} \right)^{p-1} \right)^{\frac{p-1}{p-1}}
\end{aligned}
$$

The triangle inequality applied to $\mathbb{R}^N$ with the norm $\|\cdot\|_{p-1}$ gives:

$$
\begin{aligned}
&\leq 2\left(\left(\sum_{|\alpha|\leq k}\|\partial^\alpha u\|_{L_p}^{p'(p-1)}\right)^{\frac{1}{p-1}}+\left(\sum_{|\alpha|\leq k}\|\partial^\alpha v\|_{L_p}^{p'(p-1)}\right)^{\frac{1}{p-1}}\right)^{p-1}\\
&= 2\left(\|u\|^{p'}+\|v\|^{p'}\right)^{p-1},
\end{aligned}
$$

where the last equality follows through the equations $p'(p-1)=p$ and $p-1=\frac{p}{p'}$. The remaining inequality is shown as in the case $1<p\leq 2$.

$\diamondsuit$

Next we show that if the Clarkson inequalities hold in X for some $1<p<\infty$, the Clarkson inequalities hold in the dual space for p'.

Lemma 3.27 *Let X be a Banach space such that the Clarkson inequalities (3.24) or (3.25) hold for some $1<p<\infty$. We denote $\frac{1}{p}+\frac{1}{p'}:=1$. Then the Clarkson inequalities hold also for the dual space X' for p' instead of p.*

PROOF. The result is known, cf. [31, Theorem 2.9]. We work out the details of the proof here, which hasn't been done in [31, Theorem 2.9].
For a Banach space X and $1<p<\infty$ we define the Banach space $X_p^2:=X\times X$ with the norm $\|(u_1,u_2)\|:=(\|u_1\|^p+\|u_2\|^p)^{\frac{1}{p}}$. One easily proves $(X_p^2)'=(X')_{p'}^2$.

- The case $p\geq 2$: With the transformation $\tilde{u}:=u+v$ and $\tilde{v}:=u-v$ the second inequality in (3.25) becomes

$$
\|\tilde{u}\|^p+\|\tilde{v}\|^p \geq 2^{1-p}(\|\tilde{u}+\tilde{v}\|^p+\|\tilde{u}-\tilde{v}\|^p) \tag{3.28}
$$

for all $\tilde{u}$, $\tilde{v}\in X$. By (3.28) the linear mapping $A_{p,p}:X_p^2\to X_p^2$ with

$$
A_{p,p}:(u_1,u_2)=(u_1+u_2,u_1-u_2)
$$

is continuous with $\|A_{p,p}\|\leq 2^{\frac{p-1}{p}}$ and by the first inequality in (3.25) the linear mapping $A_{p',p}:X_{p'}^2\to X_p^2$ with

$$
A_{p',p}:(u_1,u_2)=(u_1+u_2,u_1-u_2)
$$

is continuous with $\left\|A_{p',p}\right\|\leq 2^{\frac{1}{p}}$. For the adjoint mapping

$$
(A_{p,p})'=:\tilde{A}_{p',p'}:(X_p^2)'=(X')_{p'}^2\to(X_p^2)'=(X')_{p'}^2
$$

holds for every $u_1, u_2 \in X$ and every $u_1', u_2' \in X'$

$$\begin{aligned}\left\langle \tilde{A}_{p',p'}(u_1', u_2') \,\middle|\, (u_1, u_2) \right\rangle &:= \left\langle (u_1', u_2') \,\middle|\, A_{p,p}(u_1, u_2) \right\rangle \\ &= \left\langle (u_1', u_2') \,\middle|\, (u_1 + u_2, u_1 - u_2) \right\rangle \\ &= \left\langle (u_1' + u_2', u_1' - u_2') \,\middle|\, (u_1, u_2) \right\rangle \ ,\end{aligned}$$

i.e.

$$\tilde{A}_{p',p'}(u_1', u_2') = (u_1' + u_2', u_1' - u_2') \, .$$

Analogously $(A_{p',p})' =: \tilde{A}_{p',p} : (X_p^2)' = (X')_{p'}^2 \to (X_{p'}^2)' = (X')_p^2$ is given by

$$(A_{p',p})' =: \tilde{A}_{p',p}(u_1', u_2') = (u_1' + u_2', u_1' - u_2') \, .$$

One has $\left\| \tilde{A}_{p',p'} \right\| = \|A_{p,p}\| \leq 2^{\frac{p-1}{p}}$ and $\left\| \tilde{A}_{p',p} \right\| = \left\| A_{p',p} \right\| \leq 2^{\frac{1}{p}}$, cf. [59, Satz III.4.2]. The boundedness of $\tilde{A}_{p',p'}$ implies that for all $\tilde{u}', \tilde{v}' \in X'$ holds

$$\begin{aligned}\left(\left\| \tilde{u}' + \tilde{v}' \right\|^{p'} + \left\| \tilde{u}' - \tilde{v}' \right\|^{p'} \right)^{\frac{1}{p'}} &\leq 2^{\frac{p-1}{p}} \left(\left\| \tilde{u}' \right\|^{p'} + \left\| \tilde{v}' \right\|^{p'} \right)^{\frac{1}{p'}} \\ \Leftrightarrow \left\| \tilde{u}' + \tilde{v}' \right\|^{p'} + \left\| \tilde{u}' - \tilde{v}' \right\|^{p'} &\leq 2^{\frac{p'(p-1)}{p}} \left\| \tilde{u}' \right\|^{p'} + \left\| \tilde{v}' \right\|^{p'} \\ &= 2(\left\| \tilde{u}' \right\|^{p'} + \left\| \tilde{v}' \right\|^{p'}) \end{aligned} \tag{3.29}$$

and the boundedness of $\tilde{A}_{p',p}$ implies that for all $\tilde{u}', \tilde{v}' \in X'$ holds

$$\begin{aligned}\left(\left\| \tilde{u}' + \tilde{v}' \right\|^{p} + \left\| \tilde{u}' - \tilde{v}' \right\|^{p} \right)^{\frac{1}{p}} &\leq 2^{\frac{1}{p}} \left(\left\| \tilde{u}' \right\|^{p'} + \left\| \tilde{v}' \right\|^{p'} \right)^{\frac{1}{p'}} \\ \Leftrightarrow 2^{-\frac{p'}{p}} \left(\left\| \tilde{u}' + \tilde{v}' \right\|^{p} + \left\| \tilde{u}' - \tilde{v}' \right\|^{p} \right)^{\frac{p'}{p}} &\leq \left\| \tilde{u}' \right\|^{p'} + \left\| \tilde{v}' \right\|^{p'} \ . \end{aligned} \tag{3.30}$$

(3.29) gives the second inequality in (3.24) for p'. Further $1 + \frac{p'}{p} = p'$ and (3.30) give with the substitutions $\tilde{u}' =: u' + v'$ and $\tilde{v}' =: u' - v'$

$$2 \left(\left\| u' \right\|^{p} + \left\| v' \right\|^{p} \right)^{p'-1} = 2^{p' - \frac{p'}{p}} \left(\left\| u' \right\|^{p} + \left\| v' \right\|^{p} \right)^{\frac{p'}{p}} \leq \left\| u' + v' \right\|^{p'} + \left\| u' - v' \right\|^{p'} \ ,$$

which gives the second part of (3.24) for p'.

- The case $1 < p \leq 2$: With the transformation $\tilde{u} := u + v$ and $\tilde{v} := u - v$ the first inequality of (3.24) becomes with $p'(p-1) = p$

$$\|\tilde{u}\|^p + \|\tilde{v}\|^p \geq 2^{1-p} \left(\|\tilde{u} + \tilde{v}\|^{p'} + \|\tilde{u} - \tilde{v}\|^{p'} \right)^{p-1} \ . \tag{3.31}$$

So the linear mapping $A_{p,p'} : X_p^2 \to X_{p'}^2$ given by

$$A_{p,p'} : (u_1, u_2) = (u_1 + u_2, u_1 - u_2)$$

is continuous with $\left\|A_{p,p'}\right\| \leq 2^{\frac{p-1}{p}} = 2^{\frac{1}{p'}}$. By the second inequality in (3.24) the linear mapping $A_{p,p} : X_p^2 \to X_p^2$ with

$$A_{p,p} : (u_1, u_2) = (u_1 + u_2, u_1 - u_2)$$

is continuous with $\|A_{p,p}\| \leq 2^{\frac{1}{p}}$. As above the adjoint mapping

$$(A_{p,p})' =: \tilde{A}_{p',p'} : (X_p^2)' = (X')_{p'}^2 \to (X_p^2)' = (X')_{p'}^2$$

is given by

$$\tilde{A}_{p',p'}(u_1', u_2') = (u_1' + u_2', u_1' - u_2')$$

and the other adjoint mapping

$$(A_{p,p'})' =: \tilde{A}_{p,p'} : (X_{p'}^2)' = (X')_p^2 \to (X_{p'}^2)' = (X')_p^2$$

is given by

$$\tilde{A}_{p,p'}(u_1', u_2') = (u_1' + u_2', u_1' - u_2') \ .$$

As above $\left\|\tilde{A}_{p,p'}\right\| = \left\|A_{p,p'}\right\| \leq 2^{\frac{1}{p'}}$ and $\left\|\tilde{A}_{p',p'}\right\| = \|A_{p,p}\| \leq 2^{\frac{1}{p}}$. So the boundedness of $\tilde{A}_{p,p'}$ implies that for all $u', v' \in X'$ holds

$$\left(\left\|u' + v'\right\|^{p'} + \left\|u' - v'\right\|^{p'}\right)^{\frac{1}{p'}} \leq 2^{\frac{1}{p'}} \left(\left\|u'\right\|^p + \left\|v'\right\|^p\right)^{\frac{1}{p}}$$

which gives the first part of (3.25) for p'. And the boundedness of $\tilde{A}_{p',p'}$ implies that for all $u', v' \in X'$ holds

$$\begin{aligned} &\left(\left\|u' + v'\right\|^{p'} + \left\|u' - v'\right\|^{p'}\right)^{\frac{1}{p'}} \leq 2^{\frac{1}{p}} \left(\left\|u'\right\|^{p'} + \left\|v'\right\|^{p'}\right)^{\frac{1}{p'}} \\ &\Leftrightarrow \left\|u' + v'\right\|^{p'} + \left\|u' - v'\right\|^{p'} \leq 2^{p'-1} \left(\left\|u'\right\|^{p'} + \left\|v'\right\|^{p'}\right) \end{aligned}$$

Substituting $\tilde{u}' := u' + v'$ and $\tilde{v}' := u' - v'$ gives the rest of (3.25).

$\diamondsuit$

PROOF of Lemma 3.22.
By our assumptions the mapping $k \mapsto \|a'_k\|$ is not increasing. We define

$$\Delta_0 := \lim_{k\to\infty} \left\|a'_k\right\| = \inf_{k\in\mathbb{N}} \left\|a'_k\right\| .$$

Let us assume $\Delta_0 > 0$. First we observe that for $k \in \mathbb{N}$ and $\lambda \in \left]\frac{1}{2}, 1\right[$ holds

$$\begin{aligned} \left\langle b'_k \,\middle|\, j(a'_k)\right\rangle &\leq \gamma \left\|a'_k\right\| \\ \Leftrightarrow \left\langle -(1-\lambda)b'_k \,\middle|\, j(a'_k)\right\rangle &\geq -(1-\lambda)\gamma \left\|a'_k\right\| \\ \Leftrightarrow \left\langle \lambda a'_k - (1-\lambda)b'_k \,\middle|\, j(a'_k)\right\rangle &\geq (\lambda - (1-\lambda)\gamma) \left\|a'_k\right\| \ . \end{aligned} \tag{3.32}$$

It is $(\lambda - (1-\lambda)\gamma)\left\|a'_k\right\| > 0$, because $\lambda > \frac{1}{2} \geq 1 - \frac{1}{1+\gamma} = \frac{\gamma}{1+\gamma}$ which implies $0 < \lambda - (1-\lambda)\gamma$. But this gives with (3.32)

$$\left\|\lambda a'_k - (1-\lambda)b'_k\right\| \geq (\lambda(1+\gamma) - \gamma) \left\|a'_k\right\| > 0 \ . \tag{3.33}$$

We recall the Clarkson inequalities hold for the Sobolev space too, cf. Lemma 3.26. And by Lemma 3.27 we obtain similar Clarkson inequalities on the dual space of X. For all $u', v' \in X'$ hold

$$\begin{aligned} \left\|u' + v'\right\|^{p'} + \left\|u' - v'\right\|^{p'} &\leq 2(\left\|u'\right\|^{p} + \left\|v'\right\|^{p})^{p'-1} \text{ for } (p' \geq 2) \ , \\ \left\|u' + v'\right\|^{p'} + \left\|u' - v'\right\|^{p'} &\leq 2(\left\|u'\right\|^{p'} + \left\|v'\right\|^{p'}) \text{ for } (1 < p' \leq 2) \ . \end{aligned}$$

We consider the case $1 < p' \leq 2$ first. With the Clarkson inequality and (3.33) we find

$$\begin{aligned} \left\|\lambda a'_k + (1-\lambda)b'_k\right\|^{p'} &\leq 2\left(\lambda^{p'} \left\|a'_k\right\|^{p'} + (1-\lambda)^{p'} \left\|b'_k\right\|^{p'}\right) - \left\|\lambda a'_k - (1-\lambda)b'_k\right\|^{p'} \\ &\leq 2\left(\lambda^{p'} \left\|a'_k\right\|^{p'} + (1-\lambda)^{p'} L^{p'}\right) - \left\|\lambda a'_k - (1-\lambda)b'_k\right\|^{p'} \\ &\leq 2\left(\lambda^{p'} \left\|a'_k\right\|^{p'} + (1-\lambda)^{p'} \frac{L^{p'}}{\Delta_0^{p'}} \left\|a'_k\right\|^{p'}\right) - \left\|\lambda a'_k - (1-\lambda)b'_k\right\|^{p'} \\ &\leq \left(2\left[\lambda^{p'} + (1-\lambda)^{p'} \frac{L^{p'}}{\Delta_0^{p'}}\right] - (\lambda(1+\gamma) - \gamma)^{p'}\right) \left\|a'_k\right\|^{p'} \ . \end{aligned}$$

If we look at the function $\alpha : \left[\frac{1}{2}, 1\right] \to \mathbb{R}$ defined by

$$\alpha(\lambda) := 2\left[\lambda^{p'} + (1-\lambda)^{p'} \frac{L^{p'}}{\Delta_0^{p'}}\right] - (\lambda(1+\gamma) - \gamma)^{p'} \ ,$$

we observe that $\alpha(1) = 1$ and

$$\alpha'(\lambda) = 2p' \left(\lambda^{p'-1} - (1-\lambda)^{p'-1} \frac{L^{p'}}{\Delta_0^{p'}} \right) - p'(1+\gamma)(\lambda(1+\gamma)-\gamma)^{p'-1} .$$

Thus $\alpha'(1) = p'(1-\gamma) > 0$. For this reason there exists some $\lambda_0 \in \left]\frac{1}{2}, 1\right[$ with $\alpha(\lambda_0) < 1$. So we obtain

$$\left\|a'_{k+1}\right\| \leq \left\|\lambda_0 a'_k + (1-\lambda_0) b'_k\right\| \leq \alpha(\lambda_0)^{\frac{1}{p'}} \left\|a'_k\right\| .$$

Taking the limit $k \to \infty$ we obtain a contradiction to $\Delta_0 > 0$, therefore it is $\Delta_0 = 0$.
Now we consider the case $p' \geq 2$. Similar to the above we calculate with the Clarkson inequality and (3.33) and $p(p'-1) = p'$

$$\begin{aligned} & \left\|\lambda a'_k + (1-\lambda) b'_k\right\|^{p'} \\ \leq\ & 2\left(\lambda^p \left\|a'_k\right\|^p + (1-\lambda)^p \left\|b'_k\right\|^p\right)^{p'-1} - \left\|\lambda a'_k - (1-\lambda) b'_k\right\|^{p'} \\ \leq\ & \left(2\left[\lambda^p + (1-\lambda)^p \frac{L^p}{\Delta_0^p}\right]^{p'-1} - (\lambda(1+\gamma)-\gamma)^{p'} \right) \left\|a'_k\right\|^{p'} . \end{aligned}$$

Now we consider the function $\beta : \left[\frac{1}{2}, 1\right] \to \mathbb{R}$ defined by

$$\beta(\lambda) := 2\left[\lambda^p + (1-\lambda)^p \frac{L^p}{\Delta_0^p}\right]^{p'-1} - (\lambda(1+\gamma)-\gamma)^{p'} .$$

We have $\beta(1) = 1$ and $\beta'(1) = 2p(p'-1) - p'(1+\gamma) = p'(1-\gamma) > 0$, since $p(p'-1) = p'$. (To compute β' we have to consider the cases $p' = 2$ and $p' > 2$ seperately.) Therefore there exists some $\lambda_0 \in \left]\frac{1}{2}, 1\right[$ with $\beta(\lambda_0) < 1$. Thus we find again

$$\left\|a'_{k+1}\right\| \leq \left\|\lambda_0 a'_k + (1-\lambda_0) b'_k\right\| \leq \beta(\lambda_0)^{\frac{1}{p'}} \left\|a'_k\right\| .$$

Again, taking the limit $k \to \infty$, we obtain a contradiction to $\Delta_0 > 0$, therefore it is $\Delta_0 = 0$. $\diamondsuit$

Finally we extend our above algorithm to subspaces of Sobolev spaces.

Algorithm 3.34 (inner approximation for Sobolev spaces)
Let X be a subspace of a Sobolev space $W^{1,p}(\Omega)$ or any $L_p(\Omega)$ space with $1 < p < \infty$ or a Hilbert space and $f : X \to \mathbb{R}$ be locally Lipschitz continuous. Here, by $j(a'_j) \in S_X(0,1) \subset X$ we always denote the dual of $a'_j \in X'$, i.e. $\left\|j(a'_j)\right\| = 1$ and $\left\langle a'_j \,\middle|\, j(a'_j)\right\rangle = \left\|a'_j\right\|$. For fixed $k, i \in \mathbb{N}$ and fixed $x_k \in X$, $\varepsilon_{k,i} > 0$, $1 > \delta > 0$ and fixed value $h(\varepsilon_{k,i}) > 0$ do:

1. *Choose any $a'_0 \in \partial f(x_k)$ and any $\delta' \in]\delta, 1[$ and set $j = 0$.*

2. *If the inequality*

$$f\left(x_k - \varepsilon_{k,i} j(a'_j)\right) - f(x_k) \leq -\delta \left\|a'_j\right\| \varepsilon_{k,i}$$

is satisfied or

$$\left\|a'_j\right\| \leq h(\varepsilon_{k,i})$$

set $D_{k,i} = a'_j$ *and stop.*

3. *Determine by means of Algorithm 3.9 some* $\xi \in [x_k, x_k - \varepsilon_{k,i} j(a'_j)]$ *and any* $b'_j \in \partial f(\xi)$ *such that*

$$\left\langle b'_j \,\middle|\, j(a'_j) \right\rangle \leq \delta' \left\|a'_j\right\| \ .$$

4. *Choose some* $C_j \subseteq \left\{a'_l \,\middle|\, l \leq j\right\} \cup \left\{b'_l \,\middle|\, l \leq j\right\}$ *with* $a'_j, b'_j \in C_j$ *and define the* **inner approximation**

$$A_j := conv\ C_j = \overline{conv}^*\ C_j \subseteq \partial^{\varepsilon_{k,i}} f\left(x_k\right) \ .$$

5. *Compute*

$$a'_{j+1} := \arg\min \left\{ \left\|f'\right\| \mid f' \in A_j \right\} \ ,$$

increment j *by one and go to 2.*

Proposition 3.35 *In the case that* f *is Lipschitz on a neighborhood of* $B_X\left(x_k\,,\,\varepsilon_{k,i}\right)$ *and Algorithm 3.9 always terminates we obtain that Algorithm 3.34 stops after finitely many steps.*

Proof. This follows directly from Lemma 3.1 and Lemma 3.22, where L denotes the Lipschitz constant, and that the algorithm stops as soon as $\left\|a'_j\right\| \leq h(\varepsilon_{k,i})$. ♢

4 A Global Semismooth Newton Method as Specialization

We recall that the Newton method is a gradient method if we change the norm in every iteration step. In the following we discuss how this fits into our framework. We will show that under certain choices of parameters in our specialized descent Algorithm 2.38, the algorithm leads eventually into the semismooth Newton method, if the assumptions for the semismooth Newton method related to the function f are satisfied.
In the following we assume that $(X, \|\cdot\|)$ and $(Y, \|\cdot\|)$ are finite dimensional real Hilbert spaces with associated scalar products $\langle \cdot \,|\, \cdot \rangle$. W.l.o.g. we say $X = \mathbb{R}^n$ and $Y := \mathbb{R}^m$ in the whole chapter. We study functions

$$f : \mathbb{R}^n \to \mathbb{R} \quad \text{and} \quad F : \mathbb{R}^n \to \mathbb{R}^m$$

and denote by $Df(x)$ and $D^2 f(x)$ the representation of the first and second derivative of f at x with respect to the norm $\|\cdot\|$ if they exist, i.e. for all $y, z \in \mathbb{R}^n$ we require

$$\begin{aligned} \langle Df(x) \,|\, y \rangle &= f'(x)(y) \,, \\ \langle D^2 f(x)\, z \,|\, y \rangle &= \big(f''(x)(z)\big)\,(y) \,. \end{aligned}$$

The main reason for considering finite dimensional Hilbert spaces in this section is that only in this case we know how to generalize applicably the Jacobian of F.

4.1 Motivating Calculations

We now show that the Newton method is an optimal descent method, if we change in every step the norm as in Algorithm 2.38. First we consider again the problem

$$f(x) \to \min \,.$$

Let us assume for a moment that f is sufficiently smooth around the local minimizer $x_{\min}$. This means that $D^2 f(x)$ is symmetric and positively semi definite near $x_{\min}$ and in many applications it is even positively definite. (We call a symmetric matrix (strictly) positively definite in the case $\inf\limits_{\|x\|=1} \langle x \,|\, Ax \rangle > 0$. In the case $\inf\limits_{\|x\|=1} \langle x \,|\, Ax \rangle = 0$ we say the symmetric matrix A is positively semi definite.)
For a symmetric positively definite matrix A the scalar product $\langle \cdot \,|\, \cdot \rangle_A$ with

$$\langle x \,|\, y \rangle_A := \langle x \,|\, Ay \rangle \tag{4.1}$$

for all $x, y \in \mathbb{R}^n$ defines an equivalent norm $\|\cdot\|_A$ on $\mathbb{R}^n$ given by

$$\|x\|_A := \sqrt{\langle x \,|\, x \rangle_A} \tag{4.2}$$

for all $x \in \mathbb{R}^n$. Although $f'(x) \in (\mathbb{R}^n)'$ does not depend on the scalar product, the element in $\mathbb{R}^n$ which we identify with $f'(x)$ depends on the scalar product. By $D_A f(x) \in \mathbb{R}^n$ we denote that unique element of $\mathbb{R}^n$ such that for all $y \in \mathbb{R}^n$

$$\big\langle D_A f(x) \,\big|\, y \big\rangle_A := f'(x)(y) \; . \tag{4.3}$$

Since for all $y \in \mathbb{R}^n$ holds

$$\big\langle A D_A f(x) \,\big|\, y \big\rangle = \big\langle D_A f(x) \,\big|\, y \big\rangle_A = f'(x)(y) = \big\langle Df(x) \,\big|\, y \big\rangle \;\; ,$$

we obtain

$$A D_A f(x) = Df(x) \; . \tag{4.4}$$

In many applications $D^2 f(x)$ is symetric and positively definite and

$$y \mapsto \|y\|_{D^2 f(x)} = \Big\| \sqrt{D^2 f(x)}\, y \Big\|$$

is a norm for every x close to the minimizer $x_{\min}$.[17] In this case (4.4) gives

$$D_{D^2 f(x)} f(x) = D^2 f(x)^{-1} Df(x) \; .$$

This means that the optimal descent direction with respect to the norm given by $\|\cdot\|_{D^2 f(x)}$ is $-D_{D^2 f(x)} f(x) = -D^2 f(x)^{-1} Df(x)$, which is the descent direction given by the Newton method.

Next we consider a function $F : \mathbb{R}^n \to \mathbb{R}^n$ and the problem of finding a minimizer $x_{\min}$ of the function

$$\tilde{f}(x) := \frac{1}{2} \big\langle F(x) \,\big|\, F(x) \big\rangle = \frac{1}{2} \|F(x)\|^2$$

or equivalently the classical Newton problem of finding a point $x_{\min} \in \mathbb{R}^n$ such that

$$F(x) = 0 \; .$$

[17] By $\sqrt{A}$ we denote that symmetric and positively definite operator with $\sqrt{A} \cdot \sqrt{A} = A$.

We assume for a moment that F is differentiable and we denote the representation of the Jacobian at the point x with respect to the norm $\|\cdot\|$ by $DF(x)$. For $x \in \mathbb{R}^n$ it is

$$F(x+y) = F(x) + DF(x)(y) + o(\|y\|) \ .$$

Since we consider real Hilbert spaces we obtain

$$\begin{aligned} \tilde{f}(x+y) &= \tilde{f}(x) + \frac{1}{2}\left(\left\langle F(x) \,\middle|\, DF(x)(y)\right\rangle + \left\langle DF(x)(y) \,\middle|\, F(x)\right\rangle\right) + o(\|y\|) \\ &= \tilde{f}(x) + \frac{1}{2}\left(\left\langle F(x) \,\middle|\, DF(x)(y)\right\rangle + \left\langle F(x) \,\middle|\, DF(x)(y)\right\rangle\right) + o(\|y\|) \\ &= \tilde{f}(x) + \left\langle DF(x)^T F(x) \,\middle|\, y\right\rangle + o(\|y\|) \ . \end{aligned}$$

and so

$$\begin{aligned} D\tilde{f}(x) &= DF(x)^T F(x) \\ D_A\tilde{f}(x) &= A^{-1} DF(x)^T F(x) \end{aligned}$$

for every symmetric positively definite matrix A by (4.4). In the case that $DF(x)$ is invertible close to the minimizer $x_{\min}$, we have that $DF(x)^T DF(x)$ is a symmetric positively definite matrix, thus we can choose $A := DF(x)^T DF(x)$. With this choice we compute

$$D_A\tilde{f}(x) = DF(x)^{-1}(DF(x)^T)^{-1} DF(x)^T F(x) = DF(x)^{-1} F(x)$$

But this means we have again that the optimal descent direction $-D_A\tilde{f}(x)$ of $\tilde{f}$ at x with respect to the norm $\|\cdot\|_A = \sqrt{\langle \cdot \,|\, \cdot \rangle_A}$ is the classical Newton descent direction for F at the point x.

4.2 Generalized Jacobian by Clarke

As we have seen above we need the Jacobian matrix of a function F. Here we assume only that F is locally Lipschitz continuous. For this purpose we recall the generalized Jacobian by Clarke, which is a generalization of the characterization given by Proposition 1.20. We recall again that by Rademacher's Theorem F is almost everywhere differentiable.

Definition 4.5 Suppose $S \subset \mathbb{R}^n$ is a set of Lebesgue measure 0 containing all points of $\mathbb{R}^n$ in which $F : \mathbb{R}^n \to \mathbb{R}^m$ is not differentiable. Then we define the **generalized Jacobian** by

$$\partial F(x) := \operatorname{conv}\left\{ \lim_{i\to\infty} DF(x_i) \in L\left(\mathbb{R}^n; \mathbb{R}^m\right) \;\middle|\; (x_i)_{\in\mathbb{N}} \in (\mathbb{R}^n \setminus S)^{\mathbb{N}} : x_i \to x \right\} \ ,$$

where $DF(x_i) \in L\left(\mathbb{R}^n, \mathbb{R}^m\right)$ is the classical Jacobian matrix at $x_i \in \mathbb{R}^n \setminus S$.

In [13] the following three generalizations of the results for the generalized gradient are proved.

Proposition 4.6 (basic properties of the generalized Jacobian)
Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be locally Lipschitz continuous. Then we have :

1. *$\partial F(x)$ is a nonempty, convex and compact subset of $L(\mathbb{R}^n; \mathbb{R}^m)$.*

2. *∂F is closed, i.e. for all $(x_i)_{i\in\mathbb{N}}$ and $M_i \in \partial F(x_i)$ with $x_i \to: x$ and $M_i \to: M$ we have $M \in \partial F(x)$.*

3. *∂F is upper semicontinuous, i.e for all $x \in \mathbb{R}^n$ and $\varepsilon > 0$ there exists some $\delta > 0$ such that*
$$\bigcup_{y\in B_{\mathbb{R}^n}(x\,,\,\delta)} \partial F(y) \subseteq \partial F(x) + B_{L(\mathbb{R}^n,\,\mathbb{R}^m)}(0\,,\,\varepsilon)\ .$$

4. *We denote $F = (F_1, \ldots, F_m)$. If each component function F_i is Lipschitz of rank K_i at x then F is Lipschitz of rank $K = \|(K_1, \ldots, K_m)\|$ at x,*
$$\partial F(x) \subseteq \overline{B_{L(\mathbb{R}^n,\,\mathbb{R}^m)}(0\,,\,K)}$$
and
$$\partial F(x) \subseteq \prod_{1\le i\le m} \partial F_i(x)\ .$$
(By $\prod_{i=1}^{m} \partial F_i(x)$ we denote the Cartesian product of the $\partial F_i(x)$.)

Proof. The proof can be found in [13, Proposition 2.6.2]. ♢

Next we recall a generalization of the chain rule.

Proposition 4.7 (chain rule)
Let $f = g \circ F$, where $F : \mathbb{R}^n \to \mathbb{R}^m$ is Lipschitz near x and where $g : \mathbb{R}^m \to \mathbb{R}$ is Lipschitz near $F(x)$. Then f is Lipschitz near x and one has
$$\partial f(x) \subseteq conv\Big(\partial g\big(F(x)\big) \circ \partial F(x)\Big)\ . \tag{4.8}$$
If in addition g is strictly differentiable at $F(x)$, cf. Proposition 1.8, then equality holds (and conv is superfluous).

Proof. Also this proof can be found in [13, Theorem 2.6.6]. ♢

Finally we recall the generalization of Lebourg's Theorem.

Proposition 4.9 (generalized Lebourg's Theorem)
Let $F : U \to \mathbb{R}^m$ be Lipschitz on an open and convex set $U \subseteq \mathbb{R}^n$ and let $x, y \in U$. Then one has

$$F(y) - F(x) \in conv \bigcup_{s \in [x,y]} \partial F(s)(y - x) \, .$$

PROOF. The proof can be found in [13, Proposition 2.6.5] too. ♢

4.3 Semismooth Newton Method

To formulate the semismooth Newton method one has to define semismooth functions. For this reason we recall now the definition of semismooth.

Definition 4.10 Cf. [43, 46, 49, 56]. Let $U \subseteq \mathbb{R}^n$ be nonempty and open.

- The function $F : U \to \mathbb{R}^m$ is **semismooth at** $x \in U$ if it is Lipschitz continuous near x and if the following limit exists for all $\tilde{d} \in \mathbb{R}^n$:

$$\lim_{M \in \partial F(x+\sigma d): d \to \tilde{d},\, \sigma \to 0+} M \, d \, . \tag{4.11}$$

  If F is semismooth at all $x \in U$, we call F semismooth (on U).

- $F : U \to \mathbb{R}^m$ is **(one-sided) directionally differentiable** at $x \in U$ if the **(one-sided) directional derivative**

$$F'(x, \tilde{d}) := \lim_{\sigma \to 0+} \frac{F(x + \sigma \tilde{d}) - F(x)}{\sigma}$$

  exists for all $\tilde{d} \in \mathbb{R}^n$.

One easily shows the following characterization.

Proposition 4.12 (characterization of semismooth)
Let $F : U \to \mathbb{R}^m$ be defined on the open set $U \subseteq \mathbb{R}^n$. Then for every $x \in U$ the following statements are equivalent:

1. *F is semismooth at x.*
2. *Every component function of F is semismooth at x.*
3. *F is Lipschitz continuous near x, $F'(x, \cdot)$ exists and*

$$\sup_{M \in \partial F(x+y)} \big\| M \, y - F'(x, y) \big\| = o(\|y\|) \;\; as \; y \to 0 \, .$$

4. F is Lipschitz continuous near x, $F'(x,\cdot)$ exists and

$$\sup_{M\in\partial F(x+y)} \|F(x+y)-F(x)-M\,y\| = o(\|y\|) \ \text{as}\, y\to 0\ .$$

PROOF. This result is known, cf. [56, Proposition 2.7 and Proposition 2.10] and [47, Theorem 2.3.]. ♢

Further we find that the composition of semismooth functions is again semismooth.

Proposition 4.13 (composition of semismooth functions)
Let $U\subseteq\mathbb{R}^n$ and $V\subseteq\mathbb{R}^l$ be open. Further let $G:U\to V$ and $H:V\to\mathbb{R}^m$ be semismooth. Then $F=H\circ G$ is semismooth and for every $x\in U$ holds

$$F'(x,\cdot) = H'\left(G(x), G'(x,\cdot)\right)\ .$$

PROOF. This result was proved in [20, Lemma 18 and Theorem 19]. ♢

We formulate now the (inexact) Newton method for semismooth functions to find a zero of the function essentially as in [56, Algorithm 3.9]. In the following we always assume $n=m$, since we require in the theory that the elements of the generalized Jacobians are invertible. The reason why we recall the (inexact) semismooth Newton method here, is that we will gain later that under reasonable assumptions Algorithm 2.38 turns into an inexact semismooth Newton method after finitely many steps, cf. Theorem 4.31.

Algorithm 4.14 (inexact semismooth Newton method)
Let $F:\mathbb{R}^n\to\mathbb{R}^n$ be semismooth.

1. *Choose an initial point $x_0\in\mathbb{R}^n$ and set $k=0$.*
2. *Choose an approximation $\hat{M}_k\in L(\mathbb{R}^n,\mathbb{R}^m)$ of some $M_k\in\partial F(x_k)$, solve*

$$\hat{M}_k y_k = -F(x_k) \tag{4.15}$$

   and set $x_{k+1}=x_k+y_k$.
3. *If $x_{k+1}=x_k$, then stop with result $x=x_k$.*
4. *Increment k by one and go to step 2.*

Remark 4.16

- If $\hat{M}_k = M_k$ the algorithm is just called (exact) semismooth Newton method.
- In [56] the author claims to generalize the inexact semismooth Newton method to infinite dimensional Banach spaces. For this purpose his algorithm makes a further step, the so called projection step. We have reason to doubt that the results in [56] are entirely true with this projection step. Since in our applications we do not need a projection step and the Hilbert spaces are finite dimensional (approximations) we do not go into further details here and restrict to the simple version.

Before we can state a convergence result, we need three assumptions which are specializations of [56, Assumption 3.11] and [56, Assumption 3.14]. They are also the reason why we recall the theory for inexact semismooth Newton methods. Under the same assumptions we can choose the parameters in Algorithm 2.38 in such a way that we gain (superlinear) convergence too, cf. Theorem 4.31. Since those assumptions are necessary for the convergence of the inexact semismooth Newton we consider them as reasonable for Algorithm 2.38 too.

Assumption 4.17 (Regularity condition) *Let $F : \mathbb{R}^n \to \mathbb{R}^m$ be semismooth and $x \in \mathbb{R}^n$. There exists an open and bounded $U \subseteq \mathbb{R}^n$ with $x \in U$ such that for all $y \in \overline{U}$ and all $M \in \partial F(y)$ the inverse M^{-1} exists and*

$$\sup_{y \in \overline{U}} \sup_{M \in \partial F(y)} \max\left\{ \|M\| , \left\|M^{-1}\right\| \right\} =: C_M < \infty . \tag{4.18}$$

Assumption 4.19 (Dennis-Moré-type condition)
Let $(M_k)_{k\in\mathbb{N}}$ and $(\hat{M}_k)_{k\in\mathbb{N}}$ be sequences in $L(\mathbb{R}^n, \mathbb{R}^n)$ such that

$$\left\| \hat{M}_k - M_k \right\| \to 0 \text{ as } k \to \infty . \tag{4.20}$$

Corollary 4.21 *Let $F : \mathbb{R}^n \to \mathbb{R}^n$, $x \in \mathbb{R}^n$ and $U \subseteq \mathbb{R}^n$ satisfy Assumption 4.17. Then there exists constants $r > 0$ and $\hat{C}_M > 0$ such that for every*

$$\hat{M} \in \bigcup_{y \in \overline{U}} \partial F(y) + B_{L(\mathbb{R}^n, \mathbb{R}^n)}(0 , r)$$

the inverse $\hat{M}^{-1}$ exists and

$$\max\left\{ \left\|\hat{M}\right\| , \left\|\hat{M}^{-1}\right\| \right\} \leq \hat{C}_M . \tag{4.22}$$

Proof. $\bigcup_{y \in \overline{U}} \partial F(y)$ is closed by Proposition 4.6 since $\overline{U}$ is compact. Therefore Assumption 4.17 gives that $\bigcup_{y \in \overline{U}} \partial F(y)$ is compact. We denote by $GL(n, \mathbb{R})$ the general linear group.

$L(\mathbb{R}^n, \mathbb{R}^n) \setminus GL(n,\mathbb{R})$ is closed, not empty and contains no element of $\bigcup_{y\in\overline{U}} \partial F(y)$. We choose any $r > 0$ which is smaller then the distance between $\bigcup_{y\in\overline{U}} \partial F(y)$ and $L(\mathbb{R}^n, \mathbb{R}^n) \setminus GL(n,\mathbb{R})$.

$$\overline{\bigcup_{y\in\overline{U}} \partial F(y) + B_{L(\mathbb{R}^n,\mathbb{R}^n)}(0, r)} \subseteq GL(n,\mathbb{R})$$

is bounded and closed. Thus it is compact. Since the inverse operator $M \mapsto M^{-1}$ is continuous as mapping from the set $GL(n.\mathbb{R}) \subset L(\mathbb{R}^n, \mathbb{R}^n)$ into $L(\mathbb{R}^n, \mathbb{R}^n)$ we obtain the claimed boundedness condition (4.22). ♢

Corollary 4.23 *Let $F : \mathbb{R}^n \to \mathbb{R}^n$, $x \in \mathbb{R}^n$ and $U \subseteq \mathbb{R}^n$ satisfy Assumption 4.17 and choose $r > 0$ and $\hat{C}_M > 0$ according to Corrollary 4.21. Let $(x_k)_{k\in\mathbb{N}}$ be any sequence in U and choose for every $k \in \mathbb{N}$ matrices $M_k \in \partial F(x_k)$ and $\hat{M}_k \in L(\mathbb{R}^n, \mathbb{R}^n)$ with $\left\|M_k - \hat{M}_k\right\| < r$.*
Then $M_k^T M_k$ and $\hat{M}_k^T \hat{M}_k$ are symmetric, positively definite matrices for every $k \in \mathbb{N}$ by Corrollary 4.21 and the norms $\|\cdot\|_k := \|\cdot\|_{M_k^T M_k}$ and $\|\cdot\|_k := \|\cdot\|_{\hat{M}_k^T \hat{M}_k}$, cf. (4.2), satisfy the first part of Assumption 2.41 about uniform norm equivalence. The same holds true for $\|\cdot\|_k := \|\cdot\|_{M_k^T \hat{M}_k}$ in the case that the $M_k^T \hat{M}_k$ are symmetric and positively definite.

Proof. We observe first that $C_M \geq 1$ and $\hat{C}_M \geq 1$. Let $A \in L(\mathbb{R}^n, \mathbb{R}^n)$ be symmetric and positively definite. Then for every $x \in \mathbb{R}^n \setminus \{0\}$ we define $y := A^{-1}x$ and calculate

$$\begin{aligned}
\|x\|_A^2 &= \langle x \,|\, Ax\rangle \leq \|A\| \, \|x\|^2 \\
\|x\|^2 &= \langle x \,|\, Ay\rangle = \langle x \,|\, y\rangle_A \\
&\leq \|x\|_A \, \|y\|_A \\
&= \|x\|_A \sqrt{\langle A^{-1}x \,|\, AA^{-1}x\rangle} \\
&= \|x\|_A \sqrt{\langle A^{-1}x \,|\, x\rangle} \leq \|x\|_A \sqrt{\|A^{-1}\|} \, \|x\|
\end{aligned}$$

and obtain therefore

$$\|x\|_A \leq \sqrt{\|A\|} \, \|x\| \quad \text{and} \quad \|x\| \leq \sqrt{\|A^{-1}\|} \, \|x\|_A \ .$$

For every $k \in \mathbb{N}$ the choice $A := M_k^T M_k$ gives for the norm $\|\cdot\|_k := \|\cdot\|_{M_k^T M_k}$

$$\|x\|_k \leq C_M \, \|x\| \quad \text{and} \quad \|x\| \leq C_M \, \|x\|_k \ .$$

So for every $k, k' \in \mathbb{N}$ we find with $C_M \geq 1$ that

$$\|x\|_k \leq C_M \|x\| \leq C_M^2 \|x\|_{k'} \leq C_M^4 \|x\| \leq C_M^6 \|x\|_k \ .$$

So with $K_{eq} = C_M^2$ Assumption 2.41 holds for $\|\cdot\|_k := \|\cdot\|_{M_k^T M_k}$.
The choices $A := \hat{M}_k^T \hat{M}_k$ and $A := M_k^T \hat{M}_k$ give analogue the rest. ♢

Corollary 4.24 *Let $A \in L(\mathbb{R}^n, \mathbb{R}^n)$ be a symmetric and positively definite matrix. Then holds*

$$\|x\|_A \leq \sqrt{\|A\|}\, \|x\| \quad \textit{and} \quad \|x\| \leq \sqrt{\|A^{-1}\|}\, \|x\|_A \ .$$

Proof. Cf. prove of Corollary 4.23. ♢

Now we are able to prove a modified Version of [56, Theorem 3.15].

Proposition 4.25 (superlinear convergence of Newton method)
Let $F : \mathbb{R}^n \to \mathbb{R}^n$ and $x \in \mathbb{R}^n$ satisfy Assumption 4.17 with $U \subseteq \mathbb{R}^n$, $x \in U$ and $F(x) = 0$. Furthermore let $r > 0$ and $\hat{C}_M > 0$ be according to Corrollary 4.21. Then with

$$r_M := \frac{1}{2} \min \left\{ r \, , \, \frac{1}{\hat{C}_M} \right\}$$

there exists a constant r_0 (depending only on F and x) such that the following holds :
Every sequence $(x_k)_{k \in \mathbb{N}}$ with

$$\|x - x_0\| < r_0$$

produced by Algorithm 4.14, such that Assumption 4.19 is satisfied, and such that we ensure in every iteration

$$\left\| \hat{M}_k - M_k \right\| < r_M \, ,$$

converges superlinear towards x, i.e. $\frac{\|x - x_{k+1}\|}{\|x - x_k\|} \to 0$.

Proof. We provide a different proof than in [56, Theorem 3.15], because in our opinion the proof there is not complete.[18] We generalize the approach of Qi and Sun in [48, Theorem 3.2]. We utilize the idea that the classical Newton method is an application to Banach's fix point theorem.
Since U is open there exists some $r_0 > 0$ with $B_{\mathbb{R}^n}(x, r_0) \subseteq U$.
Proposition 4.12 allows to choose $r_0 > 0$ such small that for all $y \in B_{\mathbb{R}^n}(x, r_0)$ and for all

[18] In the proof of [56, Theorem 3.15] we do not see where they prove that the sequence constructed by the semismooth Newton method converges.

$M \in \partial F(y)$ holds

$$\|F(y) - F(x) - M(y-x)\| \leq \frac{1}{4\hat{C}_M} \|x - y\| \ . \tag{4.26}$$

For every $k \in \mathbb{N}$ with $x_k \in U$ it holds $\left\|\hat{M}_k^{-1}\right\| \leq \hat{C}_M$, cf. (4.22).
We observe further in the case $x_k \in U$:

$$\begin{aligned}
&\|x - x_{k+1}\| \\
&= \left\|x - x_k + \hat{M}_k^{-1} F(x_k)\right\| \\
&= \left\|\hat{M}_k^{-1}\Big((\hat{M}_k - M_k)(x - x_k) + \Big(F(x_k) - F(x) - M_k(x_k - x)\Big)\Big)\right\| \\
&\leq \left\|\hat{M}_k^{-1}\right\| \Big(\left\|\hat{M}_k - M_k\right\| \|x - x_k\| + \|F(x_k) - F(x) - M_k(x_k - x)\| \Big) \\
&\leq \hat{C}_M \Big(\left\|\hat{M}_k - M_k\right\| \|x - x_k\| + \|F(x_k) - F(x) - M_k(x_k - x)\| \Big) \qquad (4.27) \\
&\overset{(4.26)}{\leq} \hat{C}_M \Big(r_M \|x_k - x\| + \frac{1}{4\hat{C}_M} \|x_k - x\| \Big) \\
&\leq \frac{3}{4} \|x_k - x\| \ . \qquad (4.28)
\end{aligned}$$

We assume $\|x - x_0\| < r_0$, thus $x_0 \in U$. So (4.28) gives in the case $k = 0$ that

$$\|x - x_{k+1}\| \leq \frac{3}{4} \|x - x_k\| < r_0 \ . \tag{4.29}$$

Induction over k implies that $x_k \in U$ and (4.29) hold for all $k \in \mathbb{N}$. So $x_k \to x$. Thus (4.27) and the semismoothness give that the convergence is even superlinear since by Assumption 4.19

$$\hat{C}_M \left\|\hat{M}_k - M_k\right\| \to 0$$

and by Proposition 4.12

$$\frac{\|F(x_k) - F(x) - M_k(x_k - x)\|}{\|x_k - x\|} \to 0 \ . \tag{4.30}$$

♢

4.4 Superlinear Convergence of the Specialized Algorithm as Generalized Global Newton Method

We have already seen that Algorithm 2.38 converges under reasonable assumptions. We intent to choose in Algorithm 2.38 in every iteration the norm properly such that it becomes a globally convergent "generalized" inexact Newton method. We will now see that with reasonable assumptions Algorithm 2.38 becomes even in some sense a generalization of the inexact semismooth Newton method and is even superlinear convergent.

Theorem 4.31 (superlinear convergence of Algorithm 2.38)
Let $F : \mathbb{R}^n \to \mathbb{R}^n$ be a locally Lipschitz continuous function and $\|\cdot\|$ be a Hilbert space norm on $\mathbb{R}^n$. We define the function.

$$f(x) := \frac{1}{2} \|F(x)\|^2 \ . \tag{4.32}$$

We apply Algorithm 2.38 (which we call "generalized" inexact Newton method) to the function f. In every iteration step $k \in \mathbb{N}$ we choose

$$m_k := f, \quad M_k \in \partial F(x_k) \quad \text{and } \hat{M}_k \in L(\mathbb{R}^n, \mathbb{R}^n) \ ,$$

(with respect to the norm $\|\cdot\|$,) such that Assumption 4.19 is satisfied. Moreover we require $A_k := M_k^T \hat{M}_k$ to be symmetric and positively definite for every $k \in \mathbb{N}$ and therefore we can use the norm given by

$$\|x\|_k^2 := \|x\|_{A_k}^2 = \left\langle x \,\middle|\, M_k^T \hat{M}_k x \right\rangle .$$

Under these Assumptions in every iteration step k holds

$$\hat{M}_k^{-1} F(x_k) \in \partial m_k(x_k) = \partial f(x_k) \tag{4.33}$$

with respect to the norm $\|\cdot\|_k$ and the two following two statements hold independently from each other.

1. *If*

$$\sup_{k \in \mathbb{N}} \max \left\{ \|A_k\| \ , \left\|A_k^{-1}\right\| \right\} < \infty$$

   the algorithm creates a sequence with $0 \in (\partial F(x))^T F(x)$ for every accumulation point x of the sequence. If f is coercive and has only finitely many minima which are all zeros of F, then Algorithm 2.38 also converges.

2. *Now we consider Algorithm 2.38 with additionally:*[19]

   (a) *$\delta < \frac{1}{2}$,*

   (b) *we use the inner approximation Algorithm 3.3 in Step 4 of our specialized Algorithm 2.38 to determine $D_{k,i}$,*

   (c) *$h(x) < x$ in Algorithm 2.38,*

[19] The idea behind this choices is basically that we first try whether the inexact semismooth Newton direction $f_k' := \hat{M}_k^{-1} F(x_k)$ satisfies the descent step assumption (DSAs). The choices on δ, h, g and $\varepsilon_{k,0}$ basically ensure that we make a descent step in this direction in this case. If (DSAs) does not hold for $f_k' := \hat{M}_k^{-1} F(x_k)$, the algorithm chooses a more stable descent direction.

(d) in Step 3 of Algorithm 2.38 we choose concrete

$$f' := f'_k := \hat{M}_k^{-1} F(x_k)$$

$$\varepsilon_{k,0} := g(\left\|f'_k\right\|_k, \varepsilon_{k-1}) := \left\|f'_k\right\|_k \ ,$$

, cf. (4.33), if $f'_k = 0$ we stop the algorithm at the critical point x_k of f,

(e) when applying Algorithm 3.3 in the following Step 4 of Algorithm 2.38 we set in Algorithm 3.3

$$a_0 := f'_k = \hat{M}_k^{-1} F(x_k)$$

(f) and we choose $\sigma_k \geq \varepsilon_k$ such that

$$f(x_k - \sigma_k d_k) \leq f(x_k - \varepsilon_k d_k)$$

and $\frac{\sigma_k}{\varepsilon_k}$ is bounded.

We assume further Algorithm 2.38 creates a sequence $(x_k)_{k\in\mathbb{N}}$ with accumulation point x such that $F(x) = 0$ and Assumption 4.17 is satisfied for this x. Then it follows:

- *$(x_k)_{k\in\mathbb{N}} \in \mathbb{R}^n$ converges superlinear towards x.*
- *In the case that we choose addititionally $\sigma_k := \varepsilon_k$ Algorithm 2.38 becomes the inexact semismooth Newton method after finitely many steps, i.e. $x_{k+1} = x_k - \hat{M}_k^{-1} F(x_k)$ for all sufficiently large k.*[20]

Proof. We assume that the $A_k := M_k^T \hat{M}_k$ are symmetirc and positively definite. Thus for every $k \in \mathbb{N}$ exist M_k^{-1} and $\hat{M}_k^{-1}$.
We show first that

$$\hat{M}_k^{-1} F(x_k) \in \partial f(x_k)$$

if we consider on $\mathbb{R}^n$ the norm $\|\cdot\|_k$ by generalizing (4.4). By assumption $M_k \in \partial F(x_k)$ if we consider on $\mathbb{R}^n$ the norm $\|\cdot\|$ and so $M_k^T F(x_k) \in \partial f(x_k)$ if we consider on $\mathbb{R}^n$ the norm $\|\cdot\|$, cf. Proposition 4.7. Since we identify $M_k^T F(x_k)$ with $y \mapsto \left\langle M_k^T F(x_k) \,\middle|\, y \right\rangle$ we calculate for all $y \in \mathbb{R}^n$

$$\left\langle M_k^T F(x_k) \,\middle|\, y \right\rangle \ = \ \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, \hat{M}_k^T M_k y \right\rangle = \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, y \right\rangle_{A_k}$$

[20] In the prove we will see that for such $k \in \mathbb{N}$ the algorithm chooses directly $D_k = D_{k,0} = \hat{M}_k^{-1} F(x_k)$ because Algorithm 3.3 stops in Step 2 with $j = 0$ returning $D_{k,0} = a_0$.

with $\hat{M}_k^T M_k = A_k = A_k^T = M_k^T \hat{M}_k$ by assumption. But this means

$$\hat{M}_k^{-1} F(x_k) \in \partial f(x_k)$$

if we consider on $\mathbb{R}^n$ the norm $\|\cdot\|_k$.

1. Since the A_k are symmmetric, positively definite matrices by Corrollary 4.24 for every $k \in \mathbb{N}$ and every $x \in \mathbb{R}^n$ holds

$$\|x\|_k \leq \sqrt{\|A_k\|}\, \|x\| \quad \text{and} \quad \|x\| \leq \sqrt{\left\|A_k^{-1}\right\|}\, \|x\|_k \ .$$

   We denote $1 \leq \sup\limits_{k \in \mathbb{N}} \max\left\{ \|A_k\| \ , \ \left\|A_k^{-1}\right\| \right\} =: \tilde{C}$. So for every $k, k' \in \mathbb{N}$ we find that

$$\|x\|_k \leq \tilde{C}\, \|x\| \leq \tilde{C}^2\, \|x\|_{k'} \leq \tilde{C}^4\, \|x\| \leq \tilde{C}^6\, \|x\|_k \ .$$

   So with $K_{eq} = \tilde{C}^2$ the first part of Assumption 2.41 holds. Since $m_k = f$ for all $k \in \mathbb{N}$ the second part of Assumption 2.41 is satisfied too. Now Theorem 2.44 and Proposition 4.7 give $0 \in \partial f(x) = (\partial F(x))^T F(x)$ for every accumulation point x. If f is coercive and there exist only finitely many minimal points, which are all zeros, then Proposition 2.47 gives the claimed convergence.

2. We assume $F(x) = 0$ and $x_k \neq x$ for all $k \in \mathbb{N}$, since otherwise $D_k = f_k' = \hat{M}_k^{-1} F(x_k) = 0$ and the algorithm would have stoped in the critical point x.
   We choose $U \subseteq \mathbb{R}^n$, $r > 0$, C_M and $\hat{C}_M$ according to Assumption 4.17 and Corrollary 4.21. Since F is continuous and U is bounded

$$\sup_{y \in \overline{U}} \Big(\|y\| + \hat{C}_M\, \|F(y)\| \Big) =: R < \infty \tag{4.34}$$

   We denote by L the Lipschitz constant of F on $\overline{B_{\mathbb{R}^n}(0\,,\,R)}$. Since F is semismooth at $x \in U$ and U is open, there exists some $r_x > 0$ such that $B_{\mathbb{R}^n}(x\,,\,r_x) \subset U$ and for all $y \in B_{\mathbb{R}^n}(0\,,\,r_x)$ holds

$$\sup_{M \in \partial F(y)} \|M\, y - (F(y) - F(x))\| \leq \frac{\|x - y\|}{C_M} \min\left\{ \frac{1}{8}, \frac{\sqrt{1 - 2\delta}}{4 L \hat{C}_M} \right\} \tag{4.35}$$

   by Proposition 4.12 with $1 > 2\delta$. Since $1 > 2\delta$ there exists some $r_M > 0$ such that for all

$0 \leq \tilde{r} \leq r_M$ holds

$$L^2 \hat{C}_M^2 \tilde{r}^2 < \frac{1-2\delta}{16 C_M^2} < \frac{1}{8 C_M^2}\Big(1 - \delta(2 + 2\hat{C}_M\, \tilde{r})\Big)\,. \tag{4.36}$$

Next we show that the algorithm makes a descent step with

$$D_k = D_{k,0} = a_0 = \hat{M}_k^{-1} F(x_k)$$

except finitely many times.

In the following we consider such $k \in \mathbb{N}$ such that

$$x_k \in B_{\mathbb{R}^n}(x\,,\, r_x) \quad \text{and} \quad \left\|\hat{M}_k - M_k\right\| \leq \min\{r_M, r\} \tag{4.37}$$

and so by Assumption 4.17 and (4.22) hold

$$\|M_k^{-1}\| \leq C_M \quad \text{and} \quad \left\|\hat{M}_k^{-1}\right\| \leq \hat{C}_M\,. \tag{4.38}$$

Those k exist since x is an accumulation point of $(x_k)_{k\in\mathbb{N}}$ and by Assumption 4.19.
First we estimate

$$\begin{aligned}
\left\|\hat{M}_k^{-1} F(x_k)\right\|_k^2 &= \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, \hat{M}_k^{-1} F(x_k)\right\rangle_{A_k} \\
&= \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, M_k^T \hat{M}_k \hat{M}_k^{-1} F(x_k)\right\rangle \\
&= \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, (M_k^T - \hat{M}_k^T + \hat{M}_k^T) F(x_k)\right\rangle \\
&= \|F(x_k)\|^2 + \left\langle \hat{M}_k^{-1} F(x_k) \,\middle|\, (M_k^T - \hat{M}_k^T) F(x_k)\right\rangle \\
&\leq \|F(x_k)\|^2 \left(1 + \hat{C}_M \left\|M_k^T - \hat{M}_k^T\right\|\right) \\
&= f(x_k)(2 + 2\hat{C}_M \left\|M_k^T - \hat{M}_k^T\right\|)\,.
\end{aligned} \tag{4.39}$$

To simplify notation we now assume w.l.o.g. that $x = 0$. We estimate with the semismoothness of F at $x = 0 \neq x_k$ and $F(x) = 0$:

$$\begin{aligned}
f(x_k) &= \frac{1}{2}\|M_k x_k - (M_k x_k - F(x_k))\|^2 \\
&\geq \frac{1}{2}\left(\|M_k x_k\| - \|M_k x_k - F(x_k)\|\right)^2
\end{aligned}$$

$$\begin{aligned}
&\overset{(4.35)}{\geq} \frac{1}{2}\left(\left\|M_k^{-1}\right\|^{-1}\|x_k\| - \frac{1}{8}C_M^{-1}\|x_k\|\right)^2 \\
&\geq \frac{1}{2}\left(C_M^{-1}\|x_k\| - \frac{1}{8}C_M^{-1}\|x_k\|\right)^2 \\
&\geq \frac{3}{8}C_M^{-2}\|x_k\|^2 \ . \qquad (4.40)
\end{aligned}$$

For every $x_k \in U$ holds $\left\|x_k - \hat{M}_k F(x_k)\right\| \leq R$ by (4.34).
So we estimate with $\varepsilon_{k,0} := \|f_k'\|_k = \|a_0\|_k$:

$$\begin{aligned}
&f\left(x_k - \frac{\varepsilon_{k,0}}{\|a_0\|_k}a_0\right) \\
&= f(x_k - a_0) \\
&= f(x_k - \hat{M}_k^{-1}F(x_k)) \\
&= f\left(\hat{M}_k^{-1}(\hat{M}_k x_k - F(x_k))\right) \\
&= \frac{1}{2}\left\|F\left(\hat{M}_k^{-1}(\hat{M}_k x_k - F(x_k))\right)\right\|^2 \\
&\leq \frac{1}{2}L^2\left\|\hat{M}_k^{-1}\right\|^2\left\|\hat{M}_k x_k - F(x_k)\right\|^2 \\
&\leq \frac{1}{2}L^2\hat{C}_M^2\left(\left\|\hat{M}_k - M_k\right\|\|x_k\| + \|M_k x_k - F(x_k)\|\right)^2 \\
&\leq L^2\hat{C}_M^2\left(\left\|\hat{M}_k - M_k\right\|^2\|x_k\|^2 + \|M_k x_k - F(x_k)\|^2\right) \\
&\overset{(4.35)}{\leq} L^2\hat{C}_M^2\left\|\hat{M}_k - M_k\right\|^2\|x_k\|^2 + \frac{1-2\delta}{16C_M^2}\|x_k\|^2 \\
&\overset{(4.36)}{\leq} 2\frac{1-2\delta}{16C_M^2}\|x_k\|^2 \\
&\overset{(4.36)}{\leq} \frac{1}{4}C_M^{-2}\|x_k\|^2 \cdot \left(1 - \delta(2 + 2\hat{C}_M\left\|\hat{M}_k - M_k\right\|)\right) \\
&\overset{(4.40)}{<} f(x_k)\cdot\left(1 - \delta(2 + 2\hat{C}_M\left\|\hat{M}_k - M_k\right\|)\right) \quad \text{since } x_k \neq 0 \\
&\overset{(4.39)}{\leq} f(x_k) - \delta\left\|\hat{M}_k^{-1}F(x_k)\right\|_k^2 \ . \qquad (4.41)
\end{aligned}$$

We observe the equivalence

$$\begin{aligned}
&f(x_k - a_0) \ < \ f(x_k) - \delta\left\|\hat{M}_k^{-1}F(x_k)\right\|_k^2 \\
\Leftrightarrow\ &f\left(x_k - \frac{\varepsilon_{k,0}}{\|a_0\|_k}a_0\right) - f(x_k) \ < \ -\delta\|a_0\|_k\,\varepsilon_{k,0} = -\delta\left\|\hat{M}_k^{-1}F(x_k)\right\|_k^2 \ .
\end{aligned}$$

Thus Algorithm 3.3 returns a_0 in the case $\|x_k - x\| < r_x$ and $\left\|\hat{M}_k - M_k\right\| < \min\{r, r_M\}$ and so in this case $D_{k,0} = a_0$. But since

$$\|D_{k,0}\|_k = \|a_0\|_k = \varepsilon_{k,0} > h(\varepsilon_{k,0})$$

by the assumption on h, i.e. (NSAs) is not satisfied, we skip Step 5 of Algorithm 2.38 and make a descent step with $D_k = D_{k,0}$ in this case.

- We prove now the second claim, i.e. we assume $\sigma_k := \varepsilon_k$. With the choice $\sigma_k := \varepsilon_k$ we make an inexact semismooth Newton step

$$x_{k+1} = x_k - \frac{\varepsilon_{k,0}}{\|a_0\|_k} a_0 = x_k - \hat{M}_k^{-1} F(x_k)$$

  in the case $\|x_k - x\| < r_x$ and $\left\|\hat{M}_k - M_k\right\| < \min\{r, r_M\}$. We already know that the inexact semismooth Newton method converges superlinearly by Proposition 4.25. Thus if x_k is sufficiently close to x and $\left\|\hat{M}_k - M_k\right\|$ is sufficiently small, then $\|x_{k+1} - x\| \leq \|x_k - x\| < r_x$ and so $x_{k+1} \in B_{\mathbb{R}^n}(x, r_x)$ too. So by induction we obtain that if once x_k is sufficiently close to x and $\left\|\hat{M}_k - M_k\right\|$ remains sufficiently small, then the algorithm produces the same sequence as the inexact semismooth Newton method. But this is after finitely many steps the case since x is an accumulation point of $(x_k)_{k\in\mathbb{N}}$ and due to Assumption 4.19.
- It remains to show that if we choose $\sigma_k > \varepsilon_k$, c.f. Algorithm 2.38, such that $\frac{\sigma_k}{\varepsilon_k}$ is bounded with $f(x_k - \sigma_k d_k) \leq f(x_k - \varepsilon_k d_k)$ we also gain superlinear convergence. W.l.o.g. we assume again $x = 0$. Since $\frac{\sigma_k}{\varepsilon_k}$ is bounded and since by the proof of Theorem 2.44 we have $\varepsilon_k \to 0$, cf. (2.30), we obtain $\sigma_k \to 0$. So in the case that x_k is sufficiently close to $x = 0$ and k is sufficiently large it holds

$$\|x_{k+1}\| = \|x_k - \sigma_k d_k\| < \min\{r_x, R\}$$

  since $\|d_k\| = 1$. By (4.40) it is in this case:

$$\begin{aligned} \frac{3}{8} C_M^{-2} \|x_{k+1}\|^2 &\leq f(x_{k+1}) \\ &\leq f(x_k - \varepsilon_k d_k) \\ &= \frac{1}{2} \|F(x_k - \varepsilon_k d_k)\| \\ &\leq \frac{1}{2} L^2 \|x_k - \varepsilon_k d_k\|^2 \ . \end{aligned} \tag{4.42}$$

But $\tilde{x}_{k+1}^N := x_k - \varepsilon_k d_k$ would be the inexact semismooth Newton step if

$$\left\|\hat{M}_k - M_k\right\| < \min\{r, r_M\} \quad \text{and} \quad \|x_k\| < r_x \ .$$

By (4.27) and (4.30) for every $q' > 0$ there exists some $\tilde{r}_{q'} > 0$ such that $\|x_k\| < \tilde{r}_{q'}$ and $\left\|\hat{M}_k - M_k\right\| < \frac{q'}{2\hat{C}_M}$ imply

$$\left\|\tilde{x}_{k+1}^N\right\| < q' \|x_k\| \ . \tag{4.43}$$

Therefore for every fixed $q \in]0,1[$ we obtain with $q' := \frac{q}{2LC_M}$ that there exists some $0 < r_q < \min\{r_x, R\}$ such that $\|x_k\| < r_q$ and $\left\|\hat{M}_k - M_k\right\| < \frac{q'}{2\hat{C}_M}$ imply

$$\|x_{k+1}\| \overset{(4.42)}{\leq} 2LC_M \|x_k - \varepsilon_k d_k\| = 2LC_M \left\|\tilde{x}_{k+1}^N\right\| \overset{(4.43)}{<} q \|x_k\| \ .$$

Thus for those $k \in \mathbb{N}$ it is $\|x_{k+1}\| < r_q$ too. There exists some k_q such that

$$\begin{aligned} \sup_{k \geq k_q} \sigma_k &\leq \min\{r_x, R\} - r_q, \\ \sup_{k \geq k_q} \left\|\hat{M}_k - M_k\right\| &< \frac{q'}{2\hat{C}_M} \quad \text{and} \\ \left\|x_{k_q}\right\| &< r_q \end{aligned}$$

since $x = 0$ is an accumulation point of $(x_k)_{k \in \mathbb{N}}$ and due to Assumption 4.19. Thus induction gives $\|x_{k+1}\| < \min\{r_x, R\}$ and $\|x_{k+1}\| < q \|x_k\|$ for all $k > k_q$. Thus the sequence $(x_k)_{k \in \mathbb{N}}$ converges towards $x = 0$. Since $q \in]0,1[$ was arbitrary the convergence is even superlinear.

$\diamondsuit$

Remark 4.44

1. Later in this work we set $\hat{M}_k := M_k$. But in later works we hope to choose $\hat{M}_k$ is such a way that the condition of A_k improves.

2. One could weaken the assumption that for all $k \in \mathbb{N}$ the matrix A_k is symmetric and positively definite and that

$$\sup_{k \in \mathbb{N}} \max\left\{\|A_k\| \ , \ \left\|A_k^{-1}\right\|\right\} < \infty$$

by the assumption that there exists some $k_0 > 0$ such that for all $k > k_0$ the matrix A_k is symmetric and positively definite and that

$$\sup_{k>k_0} \max \left\{ \|A_k\| \, , \, \left\|A_k^{-1}\right\| \right\} < \infty \, .$$

In the case that $\|\cdot\|_{A_k}$ would be not a norm we would simply take $\|\cdot\|_k := \|\cdot\|$. We would gain the same results exept for maybe (4.33) in the case $k \leq k_0$. These results we do not formulate here to avoid technical formulations. Further we point out that the regularity and Dennis-Moré-type conditions imply that A_k is positively definite for sufficiently large k.

3. We need $A_k := M_k^T \hat{M}_k$ symmetric to gain a scalar product and a norm defining a Hilbert space. But of course if we choose $\hat{M}_k := M_k$ the matrix A_k is symmetric and positively definite, iff M_k is regular. This means the semismooth Newton method is an example, which is in the above sense a special case of Algorithm 2.38. We do not know whether the BFGS algorithm as in [1] is a special case of Algorithm 2.38 or not. The BFGS algorithm is normally formulated as a (smooth) inexact Newton method combined with efficient step size strategy applied to Df to find a minimizer of f, cf. [1]. Further we do not know whether one can formulate the BFGS algorithm without this special step size strategy or not and if in that case Assumption 4.19 is satisfied.

4. The function $F : \mathbb{R} \to \mathbb{R}$ with $F(x) := |x| + 1$ has no zeros. Algorithm 2.38 applied to the strictly convex and coercive function $f(x) := \frac{1}{2}F(x)^2$ gives a convergent sequence to the minimal point $x = 0$. But we do not expect superlinear convergence since we can not apply the theory for the inexact semismooth Newton method. Consequently we require $F(x) = 0$ in Theorem 4.31.

5 Benchmark Problems

In the following we discuss the behavior of Algorithm 2.38 in combination with the inner approach Algorithm 3.3 and compare it with other algorithms. We recall that in Algorithm 2.38 one may choose in every iteration step a different norm and a different model function m_k approximating the energy function f. We mainly consider two specializations of Algorithm 2.38.

Algorithm 5.1 (Main algorithm for benchmark problems) *Apply Algorithm 2.38 in combination with the inner approach Algorithm 3.3. We choose normally*

$$\delta = 0.35,\, \delta' = 0.3 \quad \textit{and} \quad H(x) = 0.35 \cdot x\ . \tag{5.2}$$

Further me make one of the following two specializations.

(A) We consider in every iteration step k the same norm $\|\cdot\|_k := \|\cdot\|$ (typically the Euclidean norm) and the model $m_k := f$. Thus we consider Algorithm 2.26 using Algorithm 3.3 to determine $D_{k,i}$. Further in the case that nothing else is said, we set

$$g(x,y) = y, \quad \textit{and} \quad h(x) = \frac{x}{\varepsilon_0}\ , \tag{5.3}$$

where ε_0 is the initial radius of the initial neighborhood $B_{\mathbb{R}^n}(x_0\,,\,\varepsilon_0)$.

(B) In each iteration k we take the norm $\|\cdot\|_k := \|\cdot\|_{A_k}$ for some symmetric and positively definite matrix A_k and $m_k := f$. Further we choose the functions g and h as described in Theorem 4.31 Part 2 and formally determine in each iteration point x_k some element $\tilde{f}_k \in \partial f(x_k)$ with respect to the norm $\|\cdot\|$ and set

$$f'_k := A_k^{-1}\tilde{f}'_k, \quad a_0 := f'_k, \quad \sigma_k \geq \varepsilon_k \quad \textit{and} \ \ \varepsilon_{k,0} := \left\|f'_k\right\|_k \tag{5.4}$$

as described in Theorem 4.31 Part 2 too. Note $f'_k \in \partial f(x_k)$ with respect to the norm $\|\cdot\|_k$.

Later in Section 6 we formulate an efficient Version A of Algorithm 5.1 in its entirety, cf. Algorithm 6.51. Therefore we omit to formulate the algorithms in their entirety here too. The reason why we prefer to formulate the algorithm in its entirety there is that in the case of our benchmark problems, it is quite clear how to compute an element of the generalized gradient. Those functions are strictly differentiable on an open and dense set. Thus we can give an explicit analytical formula to compute the element of the gradient at such a point, cf. Proposition 1.8. In the case that the function is not strictly differentiable at a point x, Proposition 1.18 and Proposition 1.20 give us some explicit analytical formula to compute at least some element of

the generalized gradient. (We recall that we normally only intent to find the element of the generalized gradient of the neighborhood, which has the smallest norm. Normally we do not try to compute the norm smallest element of the generalized gradient at each point.) In Section 6 we are not in such a comfortable situation. Therefore we decided to put the algorithm in its entirety there, where the difficulties are, so that we can explain them better. Moreover the goal here is just to get a first impression of the advantages and disadvantages of our algorithms.

We will look at a couple of classical benchmark problems and compare our results to the results of the bundle and the bundle trust region algorithm by W. Alt and H. Schramm, since we used their theory and algorithm for convex functions to create our algorithms for Lipschitz functions, which are not necessarily convex. We also have a look at the BFGS algorithm[21], which has been recently studied e.g. by Lewis and Overton to minimize also nonsmooth functions and it also showed promising results. We further have a short look at the gradient sampling algorithms and the genetic algorithms. Since the same benchmark functions have been used to test different algorithms, we compare the algorithms for each function. For a short description of those algorithms we refer to the beginning of Section 1.2.
We come back to the fact that all considered functions have the property that we know in every point analytically the generalized gradient. Therefore we can give for those functions always an explicit formula to compute some element of the generalized gradient. This will be not the case in later chapters where we consider functions coming from concrete applications. Of course the concrete choice of formula to compute some element of the generalized gradient has in some cases a huge effect on the computational time. To remain comparability we try to avoid at least starting points where this dependence becomes imminent clear. E.g. we try to avoid nonsmooth starting points. (If one considers such starting points it becomes important which element of the generalized gradient one takes at each iteration point. Because then the norm smallest element gives often a descent direction, which points directly to the minimizer. Thus in this case the minimization problem becomes trivial.)

5.1 Wolfe Function

The Wolfe function is a classical benchmark function. It is one of the first functions which proved that even if the function is convex and defined on a 2 dimensional Euclidean Hilbert

[21]There are different implementations of the BFGS algorithm. E.g. in Matlab exist **fminunc** and **E04KAF**. For **fminunc** is the number of iterations equal the number of gradients minus one. But for **E04KAF** is the number of function evaluations equal the number of gradients, which is different from the number of iterations, cf. [2, Section 4.8.7]. Many authors only mention the number of iterations. Since we are not always sure, which implementation they use, we quote only their iteration numbers in this case.

space, the classical steepest descent algorithm might converge to a point different from the unique minimizer. This point is even not a critial point. The Wolfe function is given by

$$f(x,y) := \begin{cases} 9x + 16|y| - x^9 & \text{if} \quad x \le 0\,, \\ 9x + 16|y| & \text{if} \quad 0 < x < |y|\,, \\ 5\sqrt{9x^2 + 16y^2} & \text{if} \quad |y| \le x\,. \end{cases} \tag{Wolfe}$$

One easily shows that the minimal point is $(-1, 0)$, where the Wolfe function attains the value -8. Steepest descent algorithms often converge to the point $(0, 0)$, cf. [1, 2], which can not be a critical point, since the Wolfe function is convex and $f(-1, 0) < f(0, 0)$. In [1, 2] various algorithms had been applied to the Wolfe functions. Their results are shown in the following table. As starting point they always take the point $(5, 4)$. Therefore we take the same starting point. We apply Algorithm 5.1 with Specialization A and $\varepsilon_0 := 0.9$. We stop the algorithm as soon as the function value is smaller than 10^{-8}.

Algorithm:	Bundle Method	Bundle Trust Region	BFGS	Alg. 5.1 Version A
Iterations:		26	21	16
Gradients:	37	37		28
$f(x_k) - f(-1, 0)$	$\approx 1.4 \cdot 10^{-10}$	$< 10^{-9}$	$< 10^{-8}$	$\approx 2.9 \cdot 10^{-12}$

Of course one could optimize the number of iterations and the number of gradients. E.g. if we consider $\delta = 0.2$, $\delta' = 0.1, \varepsilon_0 = 0.9$ and the functions $H(x) = 0.1 \cdot x$, $g(x, y) = y$, $h(x) = 0.093 \cdot \frac{x}{\varepsilon_0}$, Algorithm 5.1 with Specialization A reaches even after 12 iterations and 22 gradient computations the point $(-0.9999963746,\ -1.9 \cdot 10^{-11})$ with $f(x_k) - f(-1, 0) < 10^{-10}$. Considering other settings the algorithm needs normally $25 - 35$ gradients to get $f(x_k) - f(-1, 0) < 10^{-8}$.
We conclude with the observation that Algorithm 5.1 with Specialization A always approximates the minimizer well after only a few iterations and gradients. It appears that we need even less or not more iterations and gradients than the bundle methods and comparably many as BFGS.

5.2 q-max

Next we consider the function $f : \mathbb{R}^n \to \mathbb{R}$ with

$$f(x) := \max\left\{x_i^2 \;\middle|\; 1 \le i \le n\right\} \quad \text{for } x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n. \tag{q-max}$$

This function has been studied in [2] to show how good the bundle method and the bundle trust region method work for nonsmooth, convex functions. The author applied both algorithms to 3

different starting points, namely

$$\begin{aligned} u_+ &:= (1, 2, 3, \ldots, n) \\ v &:= 0.1 \cdot u_+ \, , \\ u_\pm &:= (u_{\pm,1}, u_{\pm,2}, \ldots, u_{\pm,n}) \, , \end{aligned}$$

where $u_{\pm,i} := i$ if $i \leq \frac{n}{2}$ and else $u_{\pm,i} := -i$.
(The author studied also the point $e := (1, \ldots, 1)$. We will not discuss e here further, since one has to be specific, which gradient one chooses for the point e, because $\lambda e \in \partial f(e)$ for some suitable $\lambda \in \mathbb{R}$ depending on n and λe is the norm smallest element of $\partial f(e)$. But λe defines a descent direction showing to the minimizer. Thus with this choice the entire problem reduces to minimizing a quadratic function on a 1 dimensional space, which is of course trivial. Unfortunately it is not clear which gradients were chosen in [2].)

5.2.1 Bundle Methods and Algorithm 5.1 Version A

We consider the bundle method (BM), the bundle trust region method (BTR) and Algorithm 5.1 with Specialization A, $\varepsilon_0 = 0.5$ and the special choice $h(x) = 15\frac{x}{\varepsilon_0}$. We stop the line search at the first point, where the functions is not decreasing.[22]
An easy computation shows that for all 3 starting points u_+, $u_\pm$ and v we gain essentially the same iteration points, except for scaling with 0.1 and changing the sign for the second half of the vectors. Therefore the function values are the same, except for multiplying by 0.01 in the case that we consider the initial value v. This we do not only expect theoretically. We also observe it in the concrete computations. Therefore we omit to give a proof.
The results for the bundle and the bundle trust region method can be found in [2].

[22] We approximate this point numerically and do not compute it analytically.

	$n = 20$			
Algorithm	**BM**		**Algorithm 5.1**	
Initial point:	u_+	$u_\pm$	u_+, $u_\pm$	v
Iterations:			142	
Gradients:	247	199	246	
Value:	$1.090 \cdot 10^{-9}$	$4.145 \cdot 10^{-9}$	$1.4 \cdot 10^{-10}$	$1.4 \cdot 10^{-12}$

	$n = 50$					
Algorithm:	**BM**		**BTR**		**Algorithm 5.1**	
Initial point:	u_+	v	u_+	v	u_+, $u_\pm$	v
Iterations:					126	
Gradients:	3108	3140	451	321	311	
Value:	95.11	0.01316	$1.3 \cdot 10^{-7}$	$9.6 \cdot 10^{-7}$	$9.6 \cdot 10^{-6}$	$9.6 \cdot 10^{-8}$

If we let in the case $n = 50$ Algorithm 5.1 compute further it produces an iteration point with the function value $\mathbf{1.9 \cdot 10^{-9}}$ after 175 iterations and **452 gradient** computations and the function value $2.0 \cdot 10^{-11}$ after 200 iterations and 537 gradient computations for the initial points u_+ and $u_\pm$. For the initial point v the function values are $1.9 \cdot 10^{-11}$ and $2.0 \cdot 10^{-13}$ respectively. Unfortunately we could only find those starting points in the literature, so we can not compare with other starting points.

5.2.2 Newton Method and Algorithm 5.1 Version B

We consider the $q-$ max function $f : \mathbb{R}^n \to \mathbb{R}$. By S we denote the set of all points at which f is not differentiable, i.e.

$$S = \left\{ x \in \mathbb{R}^n \;\middle|\; \exists i, i' \leq n \,:\, i \neq i',\; x_i = x_{i'} \text{ and } f(x) = x_i^2 \right\} .$$

Then $f \in C^\infty(\mathbb{R}^n \setminus S)$. To the function $F : \mathbb{R}^n \setminus S \to \mathbb{R}^n$ given by $F(x) := Df(x)$ we apply formally Algorithm 4.14 with $M_k = \hat{M}_k$ and stop simply in the case that $x_k \in S$. Thus we apply the standard (smooth) Newton method to f or respectively Df and stop if the iteration point is not smooth. The Hessian matrix is selfadjoint, therefore we can decompose $\mathbb{R}^n$ orthogonal into the image of the Hessian and its kernel. The Hessian matrix of f is not regular if it exists. Thus then y_k is not uniquely defined by (4.15) and we have to say which element we take. We take the unique element of the image of the Hessian, which solves (4.15). Thus we consider that Newton step, which has the smallest norm since every solution is the sum of this element and an element of the kernel. In the case that no Newton step y_k exists, we stop the algorithm too. We implemented this algorithm too. For each of the initial points u_+, v and $u_\pm$ this (Newton) algorithm stops precisely at the (global) minimizer after exactly n steps.
The Newton method is just the steepest descent method with proper step size. Thus the step

size strategy is crucial as we can also see from the fact that the bundle trust region method gives that much better results. In particular if we apply Algorithm 5.1 with Specialization B one could expect that this algorithm terminates in the minimizer after at most n steps for every starting point, if we choose the gradients again properly. In practice Algorithm 5.1 with Specialization B and $A_k := Id$, i.e. $\|\cdot\|_k = \|\cdot\|$ is the Euclidean norm, stops exactly at the minimizer after precisely n steps and n gradient computations too. This holds for both cases $n = 20$ and $n = 50$.

5.2.3 Conclusion for q-max

We observe again that Algorithm 5.1 with Specialization A gives a good approximation of the minimizer after relatively few iterations and gradient computations. Again it appears that Algorithm 5.1 is with both specializations faster than the bundle methods. Due to the special, essentially quadratic structure of the function, our basic algorithm is inferior to the Newton method (and the BFGS, which would give the same results). But knowing of the special structure of the function it is an easy task to adjust to the situation by using Algorithm 5.1 with Specialization B and gain the exact solution after only n steps too by simply choosing the parameters properly.

5.3 Rosenbrock

Now we consider the Rosenbrock function, which has the property that the path following the gradient field leads into a canyon and then the path follows this canyon. Following this canyon is for a steepest descent method very time consuming. But the Newton method and Newton based methods like the BFGS show very good performance for this function. This function is a classical benchmark problem and we have already talked about it in our basic consideration about optimal descent directions. So of course we also want to know how well Algorithm 5.1 works for this function. For $n = 2$ the function is given by

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2 \tag{Ros}$$

For $n > 2$ there exist different generalizations. There exists the coupled Rosenbrock function

$$f(x) = \sum_{i=1}^{n-1} (1 - x_i)^2 + 100(x_{i+1} - x_i^2)^2 \tag{Ros-c}$$

and the uncoupled version

$$f(x) = \sum_{i=0}^{n/2-1} (1 - x_{2i+1})^2 + 100(x_{2i+2} - x_{2i+1}^2)^2 \,. \tag{Ros-u}$$

In any case the minimal point is $(1, \ldots, 1)$ and has the value 0.

5.3.1 The Two-dimensional Case

In [1] for $n = 2$ the author chose the initial point $(-1.9, 2.0)$. He studied two versions of BFGS, the conjugated gradient method and two versions of trust region methods. For details we refer to [1].
We applied Algorithm 5.1 with Specialization A and $\varepsilon_0 := 1.5$ to the same initial point.
Taking a closer look at the function, we realize that the Hessian matrix at the point $(1, 1)$ is regular. This means we apply Algorithm 5.1 with Specialization B to $F(x) := Df(x)$, where A_k is the Hessian matrix at the iteration point x_k. One could prove similar to Theorem 4.31 superlinear convergence, what we will not do here. (Note that $\frac{1}{2} \|F(x)\|^2 \neq f(x)$ and therefor we can not apply Theorem 4.31 itself.) We are satisfied with the fact that by Proposition 2.47 the algorithm converges to the minimizer. Practically we even observe that after 12 iterations, i.e. 12 computations of the Hessian, and 20 gradient computations Algorithm 5.1 with Specialization B gives the exact solution $(1, 1)$.
The following table shows the results from [1, Section 4.10.4], the results of Algorithm 5.1 with-Specialization A after 14, 19 and 29 iterations and of Algorithm 5.1 with Specialization B.

Algorithm:	BFGS 1	BFGS 2	CG	TRM 1	TRM 2
Iteration:	24		47	31	34
Gradients:	25	49	69	93	33
Value:	$1.85 \cdot 10^{-6}$	$< 10^{-5}$	$< 10^{-3}$	$3.18 \cdot 10^{-9}$	$1.3 \cdot 10^{-15}$

Algorithm:	Algorithm 5.1 with			
	Specialization A			Specialization B
Iteration:	14	19	29	12
Gradients:	29	37	55	20
Value:	$1.74 \cdot 10^{-6}$	$2.25 \cdot 10^{-9}$	$1.26 \cdot 10^{-18}$	0.0

Again Algorithm 5.1 shows very good results. In particular the results of Specialization B are very impressive.

5.3.2 Uncoupled Rosenbrock Function

In [54] we can find results for applying the BFGS algorithm to the uncoupled Rosenbrock function for higher dimension. The author chose the initial point

$$(-1.2, 1, -1.2, 1, \ldots, -1.2, 1) .$$

One reason why the author studied this problem is that working with quasi Newton methods it appears that with increasing dimension, the accuracy of computations decreases and so the algorithm becomes slower.
This difficulty does not appear that strongly for Algorithm 5.1 with Specialization A and $\varepsilon_0 = 0.5\sqrt{\frac{n}{2}}$. The following table shows the results for the BFGS algorithm according to [54] and for the Algorithm 5.1 with Specialization A. We stop Algorithm 5.1 the first time it reaches a smaller value than in [54], after 36 iterations and for very small values.

BFGS							
$n =$	2	8	20	40	80		
Iterations:	26	41	75	130	199		
$f(x) =$	$3 \cdot 10^{-9}$	$4 \cdot 10^{-3}$	$9 \cdot 10^{-4}$	$2 \cdot 10^{-3}$	$7 \cdot 10^{-4}$.		
Algorithm 5.1 with Specialization A							
$n =$	2	8	20	40	80		
Iteration:	35	27	32	33	33		
Gradients:	71	58	67	69	145		
Value:	$1 \cdot 10^{-9}$	$2 \cdot 10^{-3}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$3 \cdot 10^{-5}$		
Algorithm 5.1 with Specialization A							
$n =$	2	8	20	40	80	200	1000
Iteration:	36	36	36	36	36	36	36
Gradients:	73	75	75	75	75	75	78
Value:	$4 \cdot 10^{-11}$	$7 \cdot 10^{-8}$	$4 \cdot 10^{-7}$	$1 \cdot 10^{-7}$	$3 \cdot 10^{-7}$	$7 \cdot 10^{-7}$	$3 \cdot 10^{-4}$
Algorithm 5.1 with Specialization A							
$n =$	2	8	20	40	80	200	1000
Iteration:	69	72	71	79	75	77	70
Gradients:	132	144	139	174	145	151	149
Value:	$6 \cdot 10^{-30}$	$2 \cdot 10^{-30}$	$4 \cdot 10^{-28}$	$4 \cdot 10^{-30}$	$2 \cdot 10^{-30}$	$1 \cdot 10^{-30}$	$3 \cdot 10^{-24}$

We observe that for small dimension Algorithm 5.1 with Specialization A appears to be slower than the BFGS algorithm. For this starting point this fact is essentially[23] independent of the setting we choose Algorithm 5.1 with Specialization A. We do not know why Algorithm 5.1

[23] Only for very special choices Algorithm 5.1 with Specialization A is faster.

with Specialization A is for this starting point slower, where as for the point $(-1.9, 2)$ the speed is similar. So we can not say whether this is coincidence due to the small numbers or not.

For large dimensions BFGS is obviously inferior.

5.3.3 Coupled Rosenbrock Function

Next we have a look at the coupled Rosenbrock function, which is often used to study linesearch with restart procedures (LSRS), genetic algorithms and particle swarm optimization algorithms, cf. [25, 38]. It has been pointed out in [25] that genetic algorithms and particle swarm optimization only work for small dimensions and that they are inferior to LSRS for higher dimensions. For this reason we compare our results only to this algorithm.
The LSRS procedure works basically as follows, for details we refer to [25].

Algorithm 5.5 (LSRS)

1. *Initialize: Choose a_0, $b_0 \in \mathbb{R}^n$ and a maximal iteration number $n_K \in \mathbb{N}$, a number $n_I \in \mathbb{N}$ of initial points and a number $n_L \in \mathbb{N}$ of line searches. Set $k = 0$.*

2. *Choose randomly $n_I \in \mathbb{N}$ initial points*

$$x_{k,i} \in [a_{k,1}, b_{k,1}] \times [a_{k,2}, b_{k,2}] \times \ldots \times [a_{k,n}, b_{k,n}] \quad \text{for} \quad i \leq n_I.$$

3. *Define (randomly or not) a (global) descent direction d_k and step sizes $\sigma_{k,l}$ for $l \leq n_L$. Try to make n_L descent steps in this direction for each iteration point $x_{k,i}$, i.e. for $l \leq n_L$ set $x_{k,i} = x_{k,i} - \sigma_{k,l} d_k$ in the case that $f(x_{k,i} - \sigma_{k,l} d_k) < f(x_{k,i})$.*

4. *Compute of the resulting points some point x_k with the minimal value, i.e.*

$$f(x_k) = \min_{i \leq n_I} f(x_{k,i}) \quad \text{and} \quad x_k \in \left\{ x_{k,i} \,\middle|\, i \leq n_I \right\} .$$

   Use x_k to define new borders a_{k+1}, $b_{k+1} \in \mathbb{R}^n$ by computing the gradient at x_k. Increment k by one and if $k < n_K$ go to Step 2.

5. *Return x_k.*

The authors studied dimensions from $n = 50$ to $n = 2000$ and took always $n_I = 500$ starting points, restarted the algorithm $n_K = 100$ times and made $n_L = 10$ line searches each time, thus

they consider 500, 000 function evaluations. Their average results for the LSRS algorithm (depending on the dimension) are shown in the following table. We couldn't find how the authors chose a_0 and b_0 either.

We compare with Algorithm 5.1 with Specialization A and $\varepsilon_0 = \sqrt{n}$. To keep the results repeatable, we define two different initial points $u,\ v \in \mathbb{R}^n$ which are given by

$$u_{2i+1} = -1.9,\ u_{2i+2} = 2.0 \quad \text{for} \quad i \geq 0\ :$$

and

$$v_{2i+1} = -1.9,\ v_{2i+2} = 0.0 \quad \text{for} \quad i \geq 0\ :$$

For the initial points u and v the results for Algorithm 5.1 with Specialization A are also shown in the following table.

LSRS					
$n =$	50	100	500	1000	2000
Gradients :	100	100	100	100	100
Function Evaluation:	500, 000	500, 000	500, 000	500, 000	500, 000
Average Value:	$1.4 \cdot 10^{-18}$	$6.9 \cdot 10^{-15}$	$2.6 \cdot 10^{-11}$	$7.4 \cdot 10^{-27}$	$2.4 \cdot 10^{-28}$
Algorithm 5.1 with Specialization A					
Initial point u:					
$n =$	50	100	500	1000	2000
Iterations:	338	390	1, 554	4, 130	8, 291
Gradients :	2, 287	2, 956	9, 974	22, 601	44, 943
Function Evaluation:	12, 750	15, 325	49, 250	118, 710	235, 984
Value:	$6.1 \cdot 10^{-29}$	$1.0 \cdot 10^{-28}$	$6.2 \cdot 10^{-30}$	$1.2 \cdot 10^{-30}$	$1.2 \cdot 10^{-29}$
Initial point v:					
$n =$	50	100	500	1000	2000
Iterations:	181	163	152	240	215
Gradients :	1, 338	1, 314	1, 438	2, 438	2, 651
Function Evaluation:	8, 069	7, 339	7, 223	10, 968	11, 456
Value:	$1.7 \cdot 10^{-29}$	$7.0 \cdot 10^{-29}$	$5.0 \cdot 10^{-30}$	$2.5 \cdot 10^{-30}$	$1.3 \cdot 10^{-30}$

We do not know the reason why the average values of the LSRS algorithm for $n = 1000$ and $n = 2000$ are much better than the values for smaller n.
The results are hard to compare. We recall that we do not know the initial boundaries a_0 and b_0. Our algorithm needs more gradients than just the 100 of LSRS. But for small dimensions, the numbers are much more precise and we need always much less function evaluations.

Although the coupled Rosenbrock function is a polynomial, thus smooth, the nonsmooth Algorithm 5.1 shows better results than classic steepest descent algorithms, which are very slow for Rosenbrock functions due to their oscillating gradients. Therefore we think that Algorithm 5.1 is an alternative to LSRS to solve Rosenbrock like functions.

5.4 Schwefel Function

Another popular example for testing stochastic algorithms like the genetic, line search with restart and particle swarm algorithms is the so called Schwefel function $f : \mathbb{R}^n \to \mathbb{R}$ which is given by

$$f(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i| \ .$$

Of course the unique minimal point is $x = 0$ and has the value 0. As for the coupled Rosenbrock function, we only consider the LSRS algorithm of [25] again. In [25] the LSRS algorithm was applied to the Schwefel function exactly the same way as for the Rosenbrock function. They gained the average values which we show in the next table.
As for the function q-max, it does not make sense to consider points of the form $\lambda(1, \ldots, 1)$ for $\lambda \in \mathbb{R}$, because depending on choice of the gradients then the problem reduces to an one dimensional problem. Therefore we consider the initial points[24]

$$u = (2.8\frac{1}{n}, 2.8\frac{2}{n}, \ldots, 2.8\frac{n}{n}) \quad \text{and} \quad v = (2.0 - 1.5\frac{1}{n}, 2.0 - 1.5\frac{2}{n}, \ldots, 2.0 - 1.5\frac{n}{n}) \ .$$

We apply again Algorithm 5.1 with Specialization A and $\varepsilon_0 = 0.015\sqrt{n}$. With this choice we obtain for u and v:

[24] We have to take care that the initial values do not become too large to handle with our computer.

LSRS					
$n =$	50	100	500	1000	2000
Gradients :	100	100	100	100	100
Function Evaluation:	$500,000$	$500,000$	$500,000$	$500,000$	$500,000$
Average Value	$1.9 \cdot 10^{-11}$	$6.9 \cdot 10^{-16}$	$4.1 \cdot 10^{-19}$	$1.1 \cdot 10^{-17}$	$3.1 \cdot 10^{-17}$
Algorithm 5.1 with Specialization A, initial point u					
$n =$	50	100	500	1000	2000
$f(x_0)$:	149.469	626.473	$1.5 \cdot 10^{8}$	$5.8 \cdot 10^{14}$	$6.0 \cdot 10^{27}$
Iterations :	166	177	220	300	329
Gradients :	392	417	631	848	1125
Function Evaluation:	$3,283$	$3,660$	$5,123$	$7,085$	$8,525$
Value :	$9.6 \cdot 10^{-21}$	$4.8 \cdot 10^{-21}$	$9.8 \cdot 10^{-21}$	$9.5 \cdot 10^{-21}$	$5.5 \cdot 10^{-21}$
Algorithm 5.1 with Specialization A, initial point v					
$n =$	50	100	500	1000	2000
$f(x_0)$:	1232.46	$2.8 \cdot 10^{6}$	$2.6 \cdot 10^{33}$	$1.3 \cdot 10^{67}$	$3.5 \cdot 10^{134}$
Iterations :	162	152	202	225	212
Gradients :	341	333	516	586	608
Function Evaluation:	$3,197$	$2,890$	$4,276$	$4,896$	$4,712$
Value :	$7.7 \cdot 10^{-21}$	$9.9 \cdot 10^{-21}$	$9.0 \cdot 10^{-21}$	$8.7 \cdot 10^{-21}$	$8.2 \cdot 10^{-21}$

We think Algorithm 5.1 with Specialization A appears to be better than LSRS for the Schwefel function, since we consider comparable many gradients, but only about one per cent of the function evaluations. Further Algorithm 5.1 finds points with smaller function values. We couldn't find smooth algorithms applied to this function. Since we do not know how the authors choose a_0 and b_0, we are satisfied with the fact that our algorithm can also solve stable this type of minimization problems and do not study this problem further.

In [25] the authors also tested LSRS for functions with several local minimizers. But we did not test Algorithm 5.1 with Specialization A to benchmark functions with several local minimizer, like the Levy and the Ackley function, cf. [25], because Algorithm 5.1 with Specialization A is designed to find local minimizers and not global. We also didn't study their quadratic function, the Sphere function and the Sum Squares function, cf. [25], since these are all quadratic functions and we already know that with the right choice of norm, our algorithm reaches the minimizer after just one iteration. And if we choose the Euclidean norm, it is well known that Newton methods and quasi Newton methods are the best choice.

5.5 Hilbert Function

To compare Algorithm 5.1 with Specialization A with Newton like algorithm we consider the Hilbert function next. We consider the function $f : \mathbb{R}^n \to \mathbb{R}$ given by

$$f(x) = x^T \cdot A \cdot x, \quad \text{with } A = \Big(\frac{1}{i+j-1}\Big)_{1\le i,j\le n} \in \mathbb{R}^{n\times n} . \tag{Hilbert}$$

A is the so called Hilbert matrix, which is very ill-conditioned. In practice however rounding errors on the representation of A probably improve the actual condition number, cf. [54]. The function is strictly convex and the unique minimal point is $x = 0$. We consider the starting point

$$x_0 = (\frac{4}{1}, \frac{4}{2}, \frac{4}{3}, \ldots, \frac{4}{n})$$

and the dimensions

$$n = 10,\ n = 40 \quad \text{and} \quad n = 80$$

as in [54].

Due to its quadratic form, this function is a perfect application for quasi Newton methods, but due to the bad condition a bad application for the Newton method. E. Spedicato applied the BFGS algorithm to this function and gained results in the following table, cf. [54].

We do not consider Algorithm 5.1 with Specialization B, because then we would only deal with the (smooth) exact Newton method, cf. Algorithm 4.14, again, which has been studied intensively for this problem. With exact computation the (smooth) exact Newton method would end up in the minimizer after just one step. But since we do not compute exactly in practice the results would only depend on the solving algorithm for linear equations, which is beyond the scope of this work. Therefore we use Algorithm 5.1 with Specialization A and $\varepsilon_0 = \sqrt{n}$, where $\|\cdot\|$ is the Euclidean norm. With this we gain the following results:

BFGS			
$n =$	10	40	80
Iterations:	13	43	83
Value:	$4 \cdot 10^{-11}$	$5 \cdot 10^{-9}$	$1 \cdot 10^{-10}$
Algorithm 5.1 with Specialization A			
$n =$	10	40	80
Iterations:	40	40	40
Gradients :	58	69	77
Value:	$7.0 \cdot 10^{-10}$	$2.2 \cdot 10^{-10}$	$3.8 \cdot 10^{-10}$
Iterations:	100	100	100
Gradients :	123	176	194
Value:	$4.6 \cdot 10^{-13}$	$3.3 \cdot 10^{-14}$	$3.0 \cdot 10^{-14}$

Frankly we are quite surprised by the still good results, which we did not expect, since the BFGS algorithm is designed for quadratic like functions. Of course the good results depend on the initial radius ε_0. The fact that the condition of A is bad, makes our surprisingly good results a bit plausible.
We know now that even for smooth functions, which have a regular Hessian matrix, Algorithm 5.1 might be still applicable with a reasonable amount of computational time. In particular for high dimensions the results are similar good. This is promising, since we want to apply Algorithm 5.1 later to "real world" problems.

5.6 Nesterov's Chebyshev-Rosenbrock Functions

As in [39] we consider the function $\tilde{f} : \mathbb{R}^n \to \mathbb{R}$ suggested by Nesterov

$$\tilde{f}(x) := \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1}(x_{i+1} - 2x_i^2 + 1)^2$$

and the nonsmooth variation

$$\hat{f}(x) := \frac{1}{4}(x_1 - 1)^2 + \sum_{i=1}^{n-1}|x_{i+1} - 2x_i^2 + 1|\,,$$

which is used to test the BFGS algorithm. For both functions the unique minimizer is given by $\overline{x} = (1, \ldots, 1)$. We consider the starting point

$$\hat{x} := (-1, 1, 1, \ldots, 1)\,.$$

5.6.1 Motivation for Definition of "Good" Approximation

Next we ask ourselves, when have we reached a sufficiently good approximation of the minimizer. Therefore we define the manifold

$$M := \left\{ x \in \mathbb{R}^n \;\middle|\; x_{i+1} = 2x_i^2 - 1,\ \ i = 1, \ldots, n-1 \right\}$$

which contains $\overline{x}$ and $\hat{x}$ too. Since for the i-th Chebyshev polynomial T_i holds $T_2 \circ T_i = T_{2i}$ and $T_2(x) = 2x^2 - 1$, we obtain for every $x = (x_1, x_2, \ldots, x_n) \in M$ with $x_1 \in [-1,\, 1]$ that

$$x_{i+1} = T_{2^i}(x_1) = \cos(2^i \arccos(x_1)) \;, \tag{Cheb-P}$$

where the representation by trigonometric functions can be found in [7]. Thus

$$M \cap [-1,\, 1]^n = \left\{ \left(x_1, \cos(2^1 \arccos(x_1)),\ \ldots,\ \cos(2^{n-1} \arccos(x_1))\ \right) \;\middle|\; x_1 \in [-1,\, 1] \right\} \;.$$

According to Lewis and Overton the BFGS algorithm generates typically iterations that approach M rapidly and then follow (not exactly) M to the minimizer. But the i-th Chebyshev polynomial oscillates $2^i - 1$ times between -1 and 1, i.e. it reaches $2^i - 1$ times both values -1 and 1. Moreover for each $i \in \mathbb{N}$ the Chebyshev polynomial T_{2^i} is monotonously increasing on $[1, \infty)$ and $T_{2^i}(1) = 1$. There exists even some $\tilde{t}^n < 1$ such that for all $i \leq n-1$ the Chebyshev polynomial T_{2^i} is positive and monotonously increasing on $(\tilde{t}^n, \infty)$, cf. the following Lemma 5.6. Typically iterations of BFGS (or Algorithm 5.1) follow the manifold very slowly as long as the manifold is wildly oscillating, but quite fast at the area where all components are monotonously increasing.
Therefore we require of a good minimizer approximation that it is close to some point of M, which does not belong to the wildly oscillating part. Good approximations should be even close to the above part, where the first component is larger than $\tilde{t}^n$. Next we give $\tilde{t}^n$ and an upper bound for the function value of such points.
Due to (Cheb-P) we can compute the largest $\tilde{t}^n < 1$ such that $T_{2^{n-1}}(\tilde{t}^n) = 0$.

Lemma 5.6 *For all $i \leq n$ the polynomial $T_{2^{i-1}}$ is positive and monotonously increasing on $(\tilde{t}^n,\ \infty)$*

Proof. Assume that not all those polynomials are positive on $(\tilde{t}^n,\ \infty)$. Note $T_{2^i} > 1$ for every $t > 1$ and $i \in \mathbb{N}$. We denote by i_0 the largest such $i < n$ for which there exists some $t_0 \in]\tilde{t}^n, 1]$ with $T_{2^{i_0-1}}(t_0) = 0$. W.l.o.g. we choose the maximal such t_0, thus $2^{i_0-1} \arccos(t_0) = -\frac{\pi}{2}$ if we define $\arccos : [-1, 1] \to [-\pi, 0]$. So $T_{2^{i_0}}(t_0) = -1$ and $T_{2^{i_0}}(1) = 1$ in contradiction to the definition of i_0 and the fact that $T_{2^{i_0}}$ is continuous. Thus all those polynomials are positive on

$(\tilde{t}^n,\ \infty)$.
This positivity, $T_{2i} = T_2 \circ T_i$ and the monotonicity of T_2 on $(0, \infty)$ give therefore the claimed monotonicity by induction. $\diamondsuit$

We denote by $\tilde{x}^n$ the vector $(\tilde{t}^n, T_2(\tilde{t}^n), \ldots, T_{2^{n-1}}(\tilde{t}^n)) \in M$. We expect of a good approximation $x = (x_1, x_2, \ldots, x_n)$ of $\overline{x}$ at least that

$$x_i > 0 \quad \text{for all } i \leq n \quad \text{and} \quad f(x) \leq f(\tilde{x}^n)\ .$$

For $n = 8$ the vector

$$\tilde{x}^8 \approx (0.99992, 0.9997, 0.9988, 0.995, 0.98, 0.92, 0.71, 0.0)$$

has a value less than $1.5 \cdot 10^{-8}$ and for $n = 10$ the vector

$$\tilde{x}^{10} \approx (0.999995, 0.99998, 0.99992, 0.9997, 0.9988, 0.995, 0.98, 0.92, 0.70, 0.0)$$

has a value less than $5.6 \cdot 10^{-12}$.
We speak of a good approximation in the case that all components are positive and that the function value is at most 10^{-15} and tacitly assume that the iterations are close to the minimizer and have left the oscillating part.

5.6.2 The Smooth Version

Indeed we observe very slow convergence rates. The following results for the BFGS algorithm are taken form [39].
We test Algorithm 5.1 with Specialization A and $\varepsilon_0 := 0.00225$, where $\|\cdot\|$ is the Euclidean norm, and the special choice $h(x) = 0.1\frac{x}{\varepsilon_0}$ and Algorithm 5.1 with Specialization B and $\varepsilon_0 = 0.5$, where A_k is the Hessian matrix at the iteration point x_k. If we take the initial point $(-1, 1, \ldots 1)$ for Algorithm 5.1 with Specialization B finishes after just one step in the minimizer, what is not interesting. Therefore we take for Algorithm 5.1 with Specialization B the little perturbed initial point $(-1.05, 1, \ldots 1)$.

$\tilde{f}$	BFGS		Algorithm 5.1 Version A		Algorithm 5.1 Version B	
Dimension:	$n = 8$	$n = 10$	$n = 8$	$n = 10$	$n = 8$	$n = 10$
Iterations:	$\approx 6,700$	$\approx 50,000$	$21,224$	$600,000$	$4,109$	$31,600$
Gradients:			$119,401$	$2,112,982$	$4,779$	$37,305$
Value :	$< 10^{-15}$	$< 10^{-15}$	$5.6 \cdot 10^{-20}$	$4.1 \cdot 10^{-7}$	0.0	$9.9 \cdot 10^{-16}$

We want to mention that for Algorithm 5.1 the results depend to a hugh extend on ε_0. It happens very easily that one takes ε_0 to large and after just a few iterations the iteration point is close to the manifold with first component larger than $\tilde{t}^n$. Thus the iterations do not go close to the wildly oscillating part. After that the iterations converge rapidly to the minimizer. E.g. with $\varepsilon_0 = 15$ Algorithm 5.1 with Specialization B creates after 19 iterations and 27 gradients a point with value less than 10^{-15}. This means we have to take unnaturally small ε_0 and force the iterations to come close to the oscillating part of M to gain comparable results. On the other hand, taking ε_0 too small leads to steepest descent algorithm like behavior. We do not know how the authors avoid this problem in [39], therefore we are very careful regarding conclusions about the speed of the BFGS algorithm compared to Algorithm 5.1, especially since the authors made clear that their computations are not made to compare with existing algorithms like the gradient sampling algorithm.
Further for Algorithm 5.1 with Specialization A the results depend on ε_0 in the sense that small perturbations might lead to a factor 2 to 10 in the amount of necessary iterations and gradients. We assume the reason for this is that the algorithm doesn't follow the manifold exactly and skips some parts of the path. Therefore we have to be careful with our results too. We didn't study this effect systematically, since the results for Algorithm 5.1 with Specialization B are already so convincing.

5.6.3 The Nonsmooth Version

Next we consider the nonsmooth function $\hat{f}$. The **BFGS algorithm** applied to $\hat{f}$, with random initial point, usually breaks down for $n \geq 4$ and produces points with value less than 10^{-8} in the case $n = 3$, cf. [39].
In contrast to the BFGS algorithm, Algorithm 5.1 with Specialization A does not break down for $n = 4$. We choose again the starting point $(-1, 1, \ldots, 1)$ and $\varepsilon_0 = 0.5$.

$\hat{f}$	Algorithm 5.1 Version A			
Dimension:	3	4	5	6
Iterations:	$4,365$	$25,766$	$219,886$	$> 1,500,000$
Gradients:	$13,691$	$106,714$	$1,124,623$	
Value:	$2.6 \cdot 10^{-15}$	$3.8 \cdot 10^{-10}$	$1.0 \cdot 10^{-8}$	$< 10^{-9}$

$n = 7$ is too much for Algorithm 5.1 too.
We point out that we only consider the sum of absolute values of numbers near to 0 in the nonsmooth case instead of the square of these numbers in the smooth case. Therefore the function

value is in the nonsmooth case larger than in the smooth case.

5.6.4 Approximating a Critical Point which is not a Minimizer

At last we take a very short look at nonsmooth function

$$f(x) = \frac{1}{4}|x_1 - 1| + \sum_{i=1}^{n-1} |x_{i+1} - 2|x_i| + 1| \ .$$

The BFGS, the gradient sampling algorithm and Algorithm 5.1 try to follow the path

$$S = \left\{ x \in \mathbb{R}^n \;\middle|\; x_{i+1} = 2|x_i| - 1 \ : \ \ i = 1, \ldots, n-1 \right\} \ ,$$

which is not a differentiable manifold. There are points of S at which the function is not regular. E.g. for $n = 2$ the point $(0, -1)$ is not regular with $0 \in \partial f(0, -1)$, but it is not a minimizer. According to [39], the gradient sampling algorithm and the BFGS algorithm converge to this point for many starting points. Algorithm 5.1 with Specialization A does this too, e.g. for the initial point $(-1, 1)$. Up to our experience, in the case $n = 2$ iterations produced by Algorithm 5.1 with Specialization A converge always to $(0, -1)$ or to the global minimizer$(1, 1)$. Since we designed Algorithm 5.1 as descent algorithm, we are surprised that the algorithm finds even a critical point, which is not a minimizer. We leave it to later work to find a way to use it in other applications.

5.6.5 Conclusion

In all situations we observe convergence to a critical point. And the convergence appears to be stable. In the smooth case we observe that Algorithm 5.1 with Specialization B is faster than BFGS, which is again faster than Algorithm 5.1 with Specialization A. However one has to be careful with the convergence speed, because it highly depends on the choice of ε_0. We had to force the algorithms to produce iterations which are close to the wildly oscillating part of the manifold. (The results also depend on the computer. The same source code produces essentially different numbers on different computers. We think the reason for that is that the iterations sometimes jump over some of single oscillations and thus skips parts of the manifold. This would give the computations a random factor.)
In the nonsmooth case we observe that Algorithm 5.1 with Specialization A is working for higher space dimensions. We can even treat the case $n = 6$ and gain stable convergence instead of just $n = 3$.
For the last function we observe convergence to critical points of which one is a minimal point

and one is not. We hope that we can use Algorithm 5.1 also to find critical points in other applications in later work.

5.7 Chebyshev Approximation by Exponential Sums

We have already mentioned the paper [4] and the gradient sampling algorithm (GS), which is based on the concept of generalized gradient on a set too. In this paper the algorithm was tested on some applied problems. To keep computations comparable we only consider the most simple problem here, which is the Chebyshev approximation by exponential sums. For a function $u : [1, 10] \to \mathbb{R}$ we consider the minimization problem of finding $a, b \in \mathbb{R}^{\frac{n}{2}}$ which minimize

$$\bar{f}(a, b) := \|u(\cdot) - \sum_{i=1}^{\frac{n}{2}} a_i e^{-b_i(\cdot)}\|_\infty \, .$$

As in [4] we first simplify this problem by replacing the supremum norm by the maximum on a finite grid with equally spaced grid points, which we explain next. We define $N = 2000$, the grid points $t_j = 1 + 9 \cdot \frac{j}{N}$ for $0 \leq j \leq N$ and consider the minimization problem of finding $a, b \in \mathbb{R}^{\frac{n}{2}}$, which minimizes

$$f(a, b) := \max_{0 \leq j \leq N} |u(t_j) - \sum_{i=1}^{\frac{n}{2}} a_i e^{-b_i t_j}| \, .$$

In [4] the author approximated the error function $\bar{f}(a, b)$ even further by the use of a one dimensional local maximization method based on successive cubic interpolation. This we do not, since it appeared not to be necessary, but it could be done theoretically. As function u we consider $u(t) = \frac{1}{t}$ too. With the smooth functions

$$h_j(a, b) := \frac{1}{t_j} - \sum_{i=1}^{\frac{n}{2}} a_i e^{-b_i t_j} \, ,$$

our minimization problem becomes the task to find $a, b \in \mathbb{R}^{\frac{n}{2}}$ which minimize

$$f(a, b) := \max_{0 \leq j \leq N} \max \{h_j(a, b), \, -h_j(a, b)\} \, . \tag{5.7}$$

By Proposition 1.18 we can explicitly determine elements of the generalized gradient when applying Algorithm 5.1 with Specialization A.
In [4] the authors gained the following results with gradient sampling algorithm. As initial point they took $a = b = 0$ and in each iteration they computed $2 \cdot n$ gradients.[25]

[25] We recall that the gradient sampling algorithm needs at least $n + 1$ gradients in each iteration.

Gradient sampling algorithm			
n	f	Iterations	Gradients
2	$8.55641 \cdot 10^{-2}$	42	168
4	$8.75226 \cdot 10^{-3}$	63	504
6	$7.14507 \cdot 10^{-4}$	166	1992
8	$5.58100 \cdot 10^{-5}$	282	4512

The authors couldn't compute the minimizer for $n > 8$, because the couldn't compute sufficiently precise.

Next we want to apply Algorithm 5.1 with Specialization A to f. But a specialty arises due to the symmetry. With the initial point $a = b = 0$ we gain that for every iteration point $x_k = (a_k, b_k)$ and every computed gradient $f' = (a', b')$ hold

$$a_{k,i} = a_{k,i+1},\ b_{k,i} = b_{k,i+1},\ a'_{k,i} = a'_{k,i+1},\ b'_{k,i} = b'_{k,i+1}$$

for all $0 < i < \frac{n}{2}$, as easy induction arguments over k show analytically. This we also observe in our computations. Therefore we omit the prove.

But this results in the fact that for every n, the sequence $(x_k)_{k\in\mathbb{N}}$ converges to the critical/saddle point $(\tilde{a}, \tilde{b})$ with

$$\begin{aligned} \tilde{a} &= \frac{2}{n}(\hat{a},\, \hat{a},\, \ldots,\, \hat{a}) \in \mathbb{R}^{\frac{n}{2}} \quad \text{and} \\ \tilde{b} &= \frac{2}{n}(\hat{b},\, \hat{b},\, \ldots,\, \hat{b}) \in \mathbb{R}^{\frac{n}{2}}\ , \end{aligned}$$

where $(\hat{a}, \hat{b}) \approx (1.43,\, 0.45) \in \mathbb{R}^2$ is the solution in the case $n = 2$. Thus like a hiker who walks along a crest downhill, the iterations of the algorithm converge into a saddle point due to the symmetry of the function f. We will see that little perturbations (or little steps left or right of the crest) lead to iterations, which do not converge into this saddle point.

To avoid this symmetry phenomena we can either perturb the initial point a little bit or we consider the similar problem

$$\hat{f}(a, b) = f(a, B\, b) \quad \text{for some } B \in \mathbb{R}^{\frac{n}{2}\times\frac{n}{2}}\ .$$

We try both approaches. We try on the on hand the initial point

$$a = -0.001 \cdot (0^2,\, 2^2, \ldots,\, (n-2)^2 \quad \text{and} \quad b = 0.001 \cdot (1^2, 3^2, \ldots, (n-1)^2)$$

and on the other hand the function:

$$\hat{f}(a,b) := \max_{0\le j\le N} \max\left\{\hat{h}_j(a,b),\ -\hat{h}_j(a,b)\right\} \to \min \tag{5.8}$$

with

$$\hat{h}_j(a,b) := \frac{1}{t_j} - \sum_{i=1}^{\frac{n}{2}} a_i e^{-ib_i t_j} \ .$$

So we substitute b_i by ib_i. This does not effect the minimal values. So we can compare our findings with those of the gradient sampling algorithm.
We apply Algorithm 5.1 with Specialization A, $\varepsilon_0 = 5\sqrt{\frac{n}{2}}$ and the special choice $H(x) = 0.1 \cdot x$. We gain the following results

Algorithm 5.1 with Specialization A						
	perturbed initial point			function $\hat{f}$		
n	$f(x_k)$	Iterations	Gradients	$\hat{f}(x_k)$	Iterations	Gradients
2	$8.55641 \cdot 10^{-2}$	10	21	$8.55641 \cdot 10^{-2}$	14	32
4	$8.75226 \cdot 10^{-3}$	44	124	$8.75226 \cdot 10^{-3}$	36	118
6	$7.14509 \cdot 10^{-4}$	95	431	$7.14509 \cdot 10^{-4}$	90	442
8	$5.57688 \cdot 10^{-5}$	406	2,547	$5.57688 \cdot 10^{-5}$	381	2,512
10	$4.24248 \cdot 10^{-6}$	2757	22,075	$4.24249 \cdot 10^{-6}$	2,766	24,066
12	$3.17295 \cdot 10^{-7}$	14.276	140.700	$3.17285 \cdot 10^{-7}$	117.329	2.529.382
14	$8.56 \cdot 10^{-7}$	17,961	180,065	$3.17570 \cdot 10^{-7}$	6,114	62,298

As we can see, Algorithm 5.1 with Specialization A also works for higher dimensions. And since we compute deliberately the gradients, we need by far less gradients. But at least for $n = 14$ we also seem to reach our computational limit, which is also due to rounding errors.

5.8 Nonlinear Regression

The following benchmark problem, a nonlinear regression problem, can be found in [1, Section 2.3.1]. For the given values

s_i	1	2	3	4	5	6	7	8	9	10
η_i	1.0	1.1	1.2	1.35	1.55	1.75	2.5	3.0	3.7	4.5

we want to minimize the function $f : \mathbb{R}^3 \to \mathbb{R}$ given by

$$f(x) = \sum_{i=1}^{10} \left(x_1 e^{s_i \cdot x_2} + x_3 - \eta_i\right)^2 \ .$$

The reason why we consider this problem is that the steepest descent method, the Nelder-Mead method and the BFGS method are not able to solve this problem for both initial points $x_0 = (0, 0, 0)$ and $x_0 = (1, 1, 1)$. All named methods stopped for at least one initial point at a point which is for sure not a minimal point. Up to our knowledge, it is not known, what is the minimal value. In [1] the author assumes that the minimal value is about 0.0861942. He gained the following results.

1. The **steepest descent method** stops for both initial points at some useless points.

2. For the initial point $x_0 = (1, 1, 1)$ the **BFGS algorithm** and the **Nelder-Mead method** stop at some useless point, which is for sure not a minimal point.
   For the initial point $x_0 = (0, 0, 0)$ the BFGS algorithm reaches after 21 iterations the point with value 0.0861942. The Nelder-Mead method is for this initial point successful too, although the last value is a bit higher.

We apply again Algorithm 5.1 with Specialization A with the Euclidean norm and Algorithm 5.1 with Specialization B where A_k is the Hessian matrix at iteration step x_k. We choose $\varepsilon_0 = 0.5$.

Algorithm:	Algorithm 5.1 Version A		Algorithm 5.1 Version B	
Initial point:	$(0, 0, 0)$	$(1, 1, 1)$	$(0, 0, 0)$	$(1, 1, 1)$
Iterations:	56	42	70	49
Gradients:	130	102	194	137
Value:	0.0861942		0.0861942	

We tested Algorithm 5.1 for other starting points. For this regression problem Algorithm 5.1 gives always the same final value at the same final iteration point, which is approximately

$$(0.270,\ 0.269,\ 0.592)\ .$$

We sum up that for this regression problem Algorithm 5.1 is slower in one point than the BFGS algorithm, but it appears stable in the sense that it appears to converge always to the minimal point. We recall that the actual minimal point is unknown. It is also interesting to see that even in the smooth case, varying the norm, i.e. using Specialization B instead of Specialization A, can lead to worse convergence results.

5.9 Conclusions for the Benchmark Problems

It appears to us that Algorithm 5.1 is stable and a promising algorithm to solve those benchmark problems. In every above situation where the other algorithms find the minimizer, Algorithm 5.1

also finds the minimizer. Algorithm 5.1 finds in even more situations the minimizer. For many benchmark problems Algorithm 5.1 needed the least iterations and gradients to come close to the minimizer. But for higher dimensions Algorithm 5.1 was always faster than the other algorithms, except for the smooth Nesterov's Chebyshev-Rosenbrock function with the given initial point.
For the above problems Algorithm 5.1 is definitely better than the bundle method, the trust region bundle method and the gradient sampling algorithm; regarding convergence speed, stability and it admits higher dimension. Algorithm 5.1 appeared to be more stable than the BFGS algorithm and gave results for higher dimensions. Regarding the speed of the algorithm, we have to say that it depends very much on the problem and even more important on the right norm. Most obvious were the differences for the smooth Nesterov's Chebyshev-Rosenbrock function, where the BFGS algorithm was faster for the given point than Algorithm 5.1 with Specialization A and Euclidean norm. But BFGS was slower than Algorithm 5.1 with Specialization B. But again we point out that neither their nor our results are made to compare convergence speed. For the nonsmooth version however only Algorithm 5.1 with Specialization A could solve the problem for higher dimensions than 3.

We can say that Algorithm 5.1 is for sure an alternative to solve nonlinear, non quadratic and nonsmooth minimization problems. In particular the stability and robustness of Algorithm 5.1 is very convincing. It even follows descent paths which might oscillate strongly. This let us hope that we can also solve practical problems, which we will do in the next section.

Let us make some first outlook for benchmark problems. Problems which we did not discuss so far, and which we leave to later work, are good stopping criterion, regulation of $\varepsilon_{k,0}$ and $\varepsilon_{k,i}$ and the related step size regulation. Especially in the choice of g, h and H lies much potential. Right now their choice is based on the try and error principle. Of course one wishes for a more analytical approach, which makes Algorithm 5.1 more universal applicable.[26]
In general many choices in the algorithm are not optimized yet. We stand just at the beginning of the development of the algorithm. The main goal of this work was to develop a stable and robust algorithm for nonsmooth functions, which is also fast for higher dimensions. Looking at the benchmark problems, we can say that we have reached this goal. Optimizing will be the next step. But one should always optimize algorithms with respect to practical problems, since in the end we want to solve those problems. Therefore we study in the next chapter the

[26] E.g. for many of the above problems slower decreasing functions than $H(x) = K \cdot x$ like $H(x) = K \cdot \sqrt{x}$ lead to better results for high dimensional problems. Using $g(x, y) > y$ lead often to better results too. We did not study this systematically, since we are mainly interested in the practical application of 1-Laplacian problem and the contact mechanics and not in gaining an optimal benchmark solver.

1-Laplacian. And come back to the above problems in the later outlook.

6 Applications to the 1-Laplace Operator

In this chapter the aim is to compute with Algorithm 2.38 first eigenfunctions of the p-Laplace operator ($p \geq 1$) for different domains in $\mathbb{R}^2$ and $\mathbb{R}^3$. The focus lies on the case $p = 1$ and $\mathbb{R}^2$, since there the solutions are often explicitly known. For this purpose we first introduce the space of bounded variation and formulate the minimization problems, which we will solve in this Chapter, cf. Section 6.1.1 and Section 6.1.2. After that we will discuss the discretization error and explain which quadrature formulae we use to implement this problem, cf. Section 6.1.3 and Section 6.1.4. Further we explain why this minimization problem is so difficult to solve and why the case $p = 1$ hasn't been studied numerically with FEM-methods yet, cf. Section 6.1.5. Next we formulate an approach to compute the first eigenvalue of the 1-Laplace operator for convex sets $\Omega \subset \mathbb{R}^2$, cf. Section 6.1.6. This approach is not based on FEM-methods, but it is an useful tool to test the results in the FEM-setting.
After this we explain in detail how we implemented Algorithm 2.38, cf. Section 6.2. Then we concentrate on the case $p = 1$ and use Algorithm 2.38 to compute first eigenfunctions of the 1-Laplace operator, cf. Section 6.3. We study intensively the case $\Omega =]0, 1[^2$ for different norms, different energy functions and different initial points, cf. Section 6.3.1. After that we study other sets $\Omega \subset \mathbb{R}^2$, cf. Section 6.4, and sets $\Omega \subset \mathbb{R}^3$, cf. Section 6.5. In the end we compare our results with results of the Literature for the case $p \geq 1.1$, cf. Section 6.7.

6.1 Introduction

In this section we will formulate the 1-Laplace eigenvalue problem, related functions and problems. Throughout this chapter Ω denotes an open and bounded subset of $\mathbb{R}^n$ with Lipschitz boundary.

6.1.1 Functions of Bounded Variation

We first introduce the space of functions of bounded variation and recall some important properties. We follow the definitions of [19, Chapter 5]. Throughout this section, U denotes an open subset of R^n. (Normally we choose $U = \mathbb{R}^n$ or $U = \Omega$.)

Definition 6.1 A function $u \in L^1(U)$ has **bounded variation** in U if

$$TV(u) := \sup \left\{ \int_U (u \, div\phi)(x)dx \;\middle|\; \phi \in C_0^1(U, \mathbb{R}^n),\ \|\phi\|_\infty \leq 1 \right\} < \infty \,. \tag{6.2}$$

$TV(u)$ is called the **total variation** of u (in U). We write

$$BV(U)$$

to denote the space of functions of bounded variation.

In [19, Section 5.1] it was pointed out that

$$W^{1,1}(\Omega) \subsetneq BV(\Omega) \subsetneq L^1(\Omega) \ .$$

Definition 6.3 A $\mathcal{L}^n$-measurable subset $C \subseteq \mathbb{R}^n$ has **finite perimeter** in Ω if

$$\chi_C \in BV(\Omega) \ . \tag{6.4}$$

Proposition 6.5 (Structure Theorem for BV Functions)
Let $u \in BV(U)$. Then there exists a Radon measure μ on U and a μ-measurable function $\sigma : U \to \mathbb{R}^n$ such that

1. *$|\sigma(x)| = 1$ μ a.e., and*
2. $$\int_U u(x)\, div\phi(x)dx = -\int_U \sigma(x) \cdot \phi(x)d\mu \quad \textit{for all } \phi \in C_0^1(U;\mathbb{R}^n) \ .$$

Proof. Cf. [19, Section 5.1]. ♢

Definition 6.6 If $u \in BV(U)$, we will henceforth write

$$|Du|$$

for the measure μ give by Proposition 6.5. The BV norm on $BV(U)$ is defined as

$$\|u\|_{BV(U)} := \|u\|_{L^1(U)} + |Du|(U) \quad \text{for every } u \in BV(U) \ .$$

$BV(\Omega)$ equipped with the BV norm is a Banach space, which is not reflexive. We will not use this norm for numerical computations, but this norm is usefull to give a trace operator on $BV(U)$.

Lemma 6.7 *There exists a bounded (with respect to the BV-norm) and linear continuation*

$$T : \ BV(\Omega) \to L^1(\partial\Omega,\, \mathcal{H}^{n-1})$$

of the **trace operator**

$$T : W^{1,1}(\Omega) \to L^1(\partial\Omega, \mathcal{H}^{n-1})$$

and for every $u \in BV(\Omega)$ *and* $\mathcal{H}^{n-1}$ *a.e.* $x \in \partial\Omega$ *holds*

$$Tu(x) = \lim_{r\to 0} \frac{1}{|B_{\mathbb{R}^n}(x\,,\,r) \cap U|} \int\limits_{B_{\mathbb{R}^n}(x\,,r)\cap U} f(y)dy\ .$$

PROOF. Cf. [19, Section 5.3] ♢

6.1.2 The p-Laplace Eigenvalue Problem

Definition 6.8 A function $u \in W_0^{1,p}(\Omega)$ $(p > 1)$ which for some λ solves

$$div\Big(|\nabla u|^{p-2}\nabla u\Big) = \lambda u$$

in the weak sense is called **eigenfunction** of the p-Laplace operator. λ is called **eigenvalue** of the p-Laplace operator.
In the case $p = 2$ we just talk of the eigenfunctions and eigenvalues of the Laplace operator.

We next consider the functions $F_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ and $G_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ for $p > 1$ given by

$$F_p(u) = \|\nabla u\|_{L_p}^p \quad \text{and} \quad G_p(u) = \|u\|_{L_p}^p$$

and the minimization problem

$$F_p(u) \to \min! \quad \text{under} \quad G_p(u) = 1 \quad \text{and } u \in W_0^{1,p}(\Omega)\ . \tag{6.9}$$

Lemma 6.10 *For* $p > 1$ *every minimizer* u *of (6.9) is a* **first eigenfunction** *of the p-Laplace operator and the minimal value* $\lambda_p := F_p(u)$ *is the* **first eigenvalue** *of the p-Laplace operator.*

PROOF. Cf. [33]. ♢

First eigenfunctions of the p-Laplace operator have been studied and computed numerically for $p > 1.1$, cf. [29, 33]. Later, we will give a roughly comparision to Algorithm 2.38, cf. Section 6.7. Typically there exists no minimizer $u \in W_0^{1,1}(\Omega)$ of (6.9) for $p = 1$. Therefore the minimization problem (6.9) is formulated on $BV(\Omega)$ in the case $p = 1$. We consider the energy function

$F_1 : BV(\Omega) \to \mathbb{R}$ given by

$$\begin{aligned} F_1(u) &:= TV(u) + \int_{\partial\Omega} |u|(x) d\mathcal{H}^{n-1}(x) \\ &= |Du|(\Omega) + \int_{\partial\Omega} |u|(x) d\mathcal{H}^{n-1}(x) \end{aligned} \tag{6.11}$$

and the minimization problem

$$F_1(u) \to \min! \quad \text{under} \quad G_1(u) := \int_{\Omega} |u| dx = 1 \quad \text{and } u \in BV(\Omega) \, . \tag{6.12}$$

For every $u \in W_0^{1,1}(\Omega)$ holds

$$F_1(u) = \|\nabla u\|_{L_1} \quad \text{and} \quad G_1(u) = \|u\|_{L_1} \, .$$

Definition 6.13 We call every minimizer u of (6.12) a **first eigenfunction** (of the 1-Laplace operator) and the minimal value $\lambda_1 := F_1(u)$ the **first eigenvalue** (of the 1-Laplace operator).

We formulated Algorithm 2.38 for unrestricted minimization problems. Therefore for $X_1 = BV(\Omega) \setminus \{0\}$ and $X_p = W_0^{1,p}(\Omega)$ for $(p > 1)$ we also look at the function $E_p : X_p \to \mathbb{R}$ given by

$$E_p(u) := \frac{F_p}{G_p}(u) \tag{6.14}$$

and the minimization problem

$$E_p(u) \to \min! \quad \text{with } u \in X_p \, . \tag{6.15}$$

Since E_p and G_p are positively p-homogenous we obtain the following.

Lemma 6.16 *Let $p \geq 1$. Every minimizer of (6.12) (or (6.9)) is a minimizer of (6.15). Moreover, if u is a minimizer of (6.15), then $\frac{u}{\|u\|_{L_1}}$ is a minimizer of (6.12) (or (6.9)).*

Later in Section 6.1.4 we also consider the function $\tilde{E}_p : X_p \to \mathbb{R}$ given by

$$\tilde{E}_p := F_p(u) + K|1 - G_p| \quad \text{for sufficiently large } K > 0 \, . \tag{6.17}$$

Theorem 8 in [33] gives that (6.12) has a minimizer, which is (up to scaling) the characteristic function of a so called Cheeger set. Therefore we now define Cheeger sets.

Definition 6.18 For a $\mathcal{L}^n$ measurable set $C \subseteq \overline{\Omega}$ we define the **perimeter** by

$$P(C) := \mathcal{H}^{n-1}(\partial C) = F_1(\chi_C) \ ,$$

cf. [19, Section 5.4], and the **area** by

$$A(C) := |C| = G_1(\chi_C) \ .$$

We consider the mapping

$$\begin{aligned} V : \left\{ C' \subset \Omega \,\middle|\, C' \text{ is open and nonempty} \right\} &\to \mathbb{R} \\ C' &\mapsto \frac{P(C')}{A(C')} = E_1(\chi_C) \ . \end{aligned}$$

We call every minimizer C of V **Cheeger set** (of Ω), cf. [32] .

In [33] a minimizer of (6.12) was constructed by showing that for each $p > 1$ a first eigenfunction of the p-Laplace operator exist and that they converge in the L_1 norm to a minimizer of (6.12) as $p \to 1$. In addition they showed that $\lambda_p \to \lambda_1$ as $p \to 1$.
The minimizer of (6.12) is not necessarily unique and not necessarily a characteristic function (up to scaling), cf. [32, Figure 5]. According to [33] for some Cheeger set C the function $\frac{1}{|C|}\chi_C$ is a minimizer of (6.12). Thus by Lemma 6.16 and the definition of Cheeger sets, $\frac{1}{|C'|}\chi_{C'}$ is a minimizer of (6.12) for every Cheeger set C'.
In the case that $\Omega \subseteq \mathbb{R}^2$ is convex, the Cheeger set is unique (up to sets of Lebesgue measure 0), cf. [32]. Figure 2 shows the Cheeger set of the square.

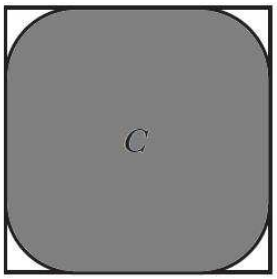

Figure 2: The Cheeger set of the square (up to Lebesgue zero sets).

The following Lemma also gives the existence of a minimizer of (6.12). We will apply this Lemma in Section 6.1.3.

Lemma 6.19

1. *The total variation TV and F_1 are lower semicontinuous in the case that we consider on $BV(\Omega)$ the L^1-norm.*

2. *Assume $(u_k)_{k\in\mathbb{N}}$ is a sequence in $BV(\Omega)$ satisfying*

$$\sup_{k\in\mathbb{N}} \|u_k\|_{BV(\Omega)} = \sup_{k\in\mathbb{N}} \Big(\|u_k\|_{L_1(\Omega)} + TV(u_k) \Big) < \infty \, .$$

Then there exists a subsequence $(u_{k(i)})_{i\in\mathbb{N}}$ and a function $u \in BV(\Omega)$ such that

$$u_{k(i)} \to u$$

in $L^1(\Omega)$ as $i \to \infty$.

Proof.

1. Theorem 1 in [19, Section 5.2] yields that TV is lower semicontinuous with respect to the L^1-norm.
   For every $u \in BV(\Omega)$ we denote by $\overline{u} \in L^1(\mathbb{R}^n)$ the continuation of u such that $u(x) = 0$ for every $x \notin \Omega$. Theorem 1 in [19, Section 5.4] gives that $\overline{u} \in BV(\mathbb{R}^n)$ and

$$F_1(u) = TV(\overline{u}) \, .$$

   Further the mapping $BV(\Omega) \to BV(\mathbb{R}^n)$ with $u \mapsto \overline{u}$ is continuous with respect to the L^1-norms. Thus we can apply Theorem 1 in [19, Section 5.2] again and gain that F_1 is lower semicontinuous with respect to the L^1-norm too.

2. Theorem 4 in [19, Section 5.2] gives the remaining claim.

♢

In the next section we discuss approximation errors. We distinguish between three types of errors: the discretization error, the error coming from the quadrature formula and the rounding error. (Under rounding error we understand the error that arise from computing with approximated numbers instead of their exact mathematical value.) A detailed discussion of rounding errors is beyond the scope of this work. We discuss the error through the quadrature formula in Section 6.1.4.

6.1.3 The Discretrization Error

Next we discuss the discretization error for $p = 1$. For the well known case $p > 1$ we refer to the literature, cf. [11].

We recall that at least one minimizer of (6.12) is (up to scaling) the characteristic function of a Cheeger set. Often the minimizer of (6.12) is unique (up to to the sign). E.g. in the case that $\Omega = [0,1]^2$ this is the case. This makes it very difficult to approximate the solution with finite element methods.

Piecewise Constant Functions

As a first idea one might think of approximating the first eigenfunction u of the 1-Laplace operator by piece wise constant functions u_a. These functions u_a are of typically of bounded variation and it is quite easy to determine the values $F_1(u_a)$ and $G_1(u_a)$. But with common meshes there is no hope to approximate well the minimal value $F_1(u)$ by values $F_1(u_a)$, where u_a is a piecewise constant function on the fixed mesh. Typically the Cheeger set C has a boundary ∂C which is partially curved. Thus the mesh has to reflect the curvature of ∂C. This turns out to be difficult for fixed orientated meshes, since ∂C is typically unknown.

Let us for example consider for $\Omega \subset \mathbb{R}^2$ a mesh, consisting only of rectangles with edges parallel to the axes. It is easily shown that for every rectangle $R \supseteq \Omega$, with edges parallel to the axes too and every $v \in L^1(\Omega)$, which is constant on every rectangle of the mesh, it holds

$$\frac{P(R)}{A(R)} \leq \frac{F_1(v)}{G_1(v)}.$$

Thus in the simple case that $\Omega = [0,1]^2$ one has, no matter how fine the mesh is, that if the mesh consists only of rectangles with edges parallel to the axes of Ω,

$$4 = \frac{P(\Omega)}{A(\Omega)} \leq \frac{F_1(v)}{G_1(v)}$$

for any $v \in L^1(\Omega)$ which is constant on every rectangle of the mesh. But we will see later, the minimal value of (6.12) for $\Omega = [0,1]^2$ is $F_1(u) \approx 3.77$, cf. Equation (6.54).

Taking uniformly orientated triangles instead of rectangles, leads to similar problems. It is an open question, if one can approximates the first eigenfunction by piecewise constant functions on a fixed orientated mesh by refining the mesh.

In special cases it is possible to approximate well the minimizer of (6.12) by adapting the mesh in every iteration, cf. Section 6.1.6. We are looking for a generally applicable approach and therefore choose another ansatz.

Continuous, Piecewise Affine Functions

Our ansatz is to approximate the BV functions by continuous and piecewise affine functions, which are 0 at the boundary. These functions are in the Sobolev spaces $W_0^{1,p}(\Omega)$ for $p \geq 1$. The

following Lemma motivates this ansatz.

Lemma 6.20 *Let $\Omega \subseteq \mathbb{R}^n$ be open and bounded with Lipschitz boundary and let additionally $u \in BV(\Omega) \cap L^p(\Omega)$ for some $p \in [1, \infty)$. Then there exists a sequence $(u_k)_{k\in\mathbb{N}}$ in $C_0^\infty(\Omega)$ such that for any $q \in [1, p]$,*

$$u_k \to u$$

in $L^q(\Omega)$ and

$$\int_\Omega |\nabla u_k(x)| dx = F_1(u_k) \to F_1(u) \,.$$

Proof. Cf. [40, Lemma 3.2]. ♢

Since in our applications Ω satisfies the assumptions of Lemma 6.20 and some minimizing function u of (6.12) is bounded, thus $u \in L^p(\Omega)$ for every $p \in [1, \infty]$, we can apply Lemma 6.20. Hence there exists a sequence $(u_k)_{k\in\mathbb{N}}$ in $C_0^\infty(\Omega)$ such that for every $q \in [1, \infty]$

$$|F_1(u) - F_1(u_k)| \to 0 \,, \quad \text{and} \quad \|u - u_k\|_{L^q} \to 0 \quad \text{as} \quad k \to \infty \,.$$

Since the 1950th, there has been a large theory developed to approximate elements $u_k \in C_0^\infty(\Omega)$ by finite elements with respect to a Sobolev norm, cf. [11]. Further there have been several good programs implemented to compute these approximations. This theory and these programs will not be discussed in detail. The finite element software "FreeFEM++" is used to implement Algorithm 2.38.

We know that the continuous and piecewise affine functions which are 0 at the boundary are dense in $W_0^{1,1}(\Omega)$, cf. [11]. Since for all $v \in W_0^{1,1}(\Omega)$ holds $F_1(v) = \|\nabla v\|_{L_1}$ we obtain that F_1 and G_1 restricted to $W_0^{1,1}(\Omega)$ are Lipschitz continuous with respect to the Sobolev norm.

Next we choose a sequence of finite dimensional subspaces X_k of $W_0^{1,1}(\Omega)$ such that the projection operators

$$\mathcal{P}_k : W_0^{1,1}(\Omega) \to X_k \subset W_0^{1,1}(\Omega)$$

converge towards the identity with respect to the operator norm as $k \to \infty$. Such a sequence of spaces $(X_k)_{k\in\mathbb{N}}$ exists for $\Omega \subseteq \mathbb{R}^n$ with $n \leq 3$, cf. [11, Chapter III Theorem 16.2] or [21].

We denote by u a minimizer of (6.12) and choose for $p = 1$ a sequence $(u_k)_{k\in\mathbb{N}}$ according to Lemma 6.20. This sequence is bounded in $W_0^{1,1}(\Omega)$ by definition. Further we consider a sequence $(v_k)_{k\in\mathbb{N}}$ in X_k of minimizers of (6.12) restricted to X_k. ($\left\{\tilde{u} \in X_k \,\middle|\, G_1(\tilde{u}) = 1\right\}$ is compact and F_1 restricted to $X_k \subset W_0^{1,1}(\Omega)$ is continuous. Thus a minimizer of (6.12) restricted to X_k exists.)

We observe

$$F_1(u) \leq F_1(v_k) \leq F_1\left(\mathcal{P}_k(u_k)\right) \leq F_1(u_k) + L\,\|\mathcal{P}_k(u_k) - u_k\|_{W_0^{1,p}(\Omega)} \to F_1(u) \,,$$

where L denotes the Lipschitz constant of F_1,and

$$G_1(v_k) = 1 \to 1 = G_1(u) \, .$$

Hence, Lemma 6.19 yields that every subsequence of $(v_k)_{k\in\mathbb{N}}$ has a subsequence which converges to some $v \in BV(\Omega)$ with respect to the L^1-norm such that $F_1(v) \leq F_1(u)$ and $G_1(v) = 1$. By definition of u it follows $F_1(u) = F_1(v)$. Thus v is a minimizer of (6.12) too. In many applications the minimizer u of (6.12) is unique up to the sign. In this case $v = \pm u$. Without loss of generality $u = \frac{1}{|c|}u \geq 0$. Then the subsequence principle gives that $|v_k| \to u$ in $L^1(\Omega)$.
We sum up:

Proposition 6.21 *Suppose we choose a sequence of finite dimensional subspaces X_k of $W_0^{1,1}(\Omega)$ such that the projection operators*

$$\mathcal{P}_k : W_0^{1,1}(\Omega) \to X_k \subset W_0^{1,1}(\Omega)$$

converge towards the identity with respect to the operator norm as $k \to \infty$. Let $(v_k)_{k\in\mathbb{N}}$ a sequence of minimizers of (6.12) restricted to X_k.

1. *Every subsequence of $(v_k)_{k\in\mathbb{N}}$ has a subsequence which converges to some minimizer of (6.12) with respect to the L^1-norm.*
2. *If the minimizer u of (6.12) is unique (up to the sign), then $|v_k| \to u$ in $L^1(\Omega)$ or $|v_k| \to -u$ in $L^1(\Omega)$.*

To our knowledge, nobody has investigated the convergence order so far. We also do not expect any results due to the complexity of the problem.

In the following we will always solve (6.12) restricted to some fixed and finitely dimensional subspace X_k. Typically X_k is given by a subset of the continuous and piecewise affine functions on Ω.

6.1.4 Quadrature Formulae and the Related Errors

In this section we describe the quadrature formulae to compute expressions like $E_p(u)$ or $\tilde{E}_p(u)$ with $p \geq 1$ for some continuous and piecewise affine function $u \in BV(\Omega)$. Further we specify the choice of elements of $\partial E_p(u)$ and $\partial \tilde{E}_p(u)$ for these u.
To keep the notation simple we only discuss exemplarily the case $n = 2$. The case $n = 3$ can

be treated similar. We have implemented and tested Algorithm 2.38 for $n = 2$ and $n = 3$, cf. Section 6.4 and Section 6.5.

We denote by $\Omega \subseteq \mathbb{R}^2$ a fixed open and bounded set with Lipschitz boundary and by τ_h a given mesh of regular triangles T_h such that $\bigcup\limits_{T_h \in \tau_h} \overline{T_h} = \overline{\Omega_h} \approx \Omega$ and $\bigcup\limits_{T_h \in \tau_h} T_h \subseteq \Omega$. We do not assume that the mesh is uniform. To create the mesh we typically define a continuous function $\Phi : I \to \partial\Omega$, where $I \subset \mathbb{R}$ is a finite union of intervals. Then we use the standard routines given in FreeFEM++ to create a mesh. By V_h we denote the space of continuous functions which are affine on every triangle $T_h \in \tau_h$ and which are 0 on the boundary of $\bigcup\limits_{T_h \in \tau_h} \overline{T_h}$.

First, we treat the functions $F_p : V_h \to \mathbb{R}$ and $G_p : V_h \to \mathbb{R}$ for $p \geq 1$. In the following we will show that there exist quadrature formulae which are exact for $F_p(v)$ with $p \geq 1$ for every $v \in V_h$. These formulae are also exact for $G_1(v)$ for every $v \in V_h$ with $v \geq 0$ or $v \leq 0$. But they are (typically) not exact for $G_p(v)$ for $1 < p \neq 2$ and $v \in V_h$. These quadrature formulae are implemented in FreeFEM++ and we will use them tacitly in our later computations. By using the quadrature formulae we approximate G_p by some function $G_p^a : V_h \to \mathbb{R}$ for $p \geq 1$. Further we will even see that F_p and G_p^a are continuously differentiable on an open and dense subset of $D \subset V_h$ for $p \geq 1$.
We observe that for every triangle $T_h \in \tau_h$ with

$$T_h = conv\,\{a_{T_h},\, b_{T_h},\, c_{T_h}\} \tag{6.22}$$

there exists an unique affine bijective mapping $A_{T_h} : \mathbb{R}^2 \to \mathbb{R}^2$ such that

$$A_{T_h}(a_{T_h}) = (0,0),\ A_{T_h}(b_{T_h}) = (1,0) \quad \text{and} \quad A_{T_h}(c_{T_h}) = (0,1)\ .$$

Thus A_{T_h} maps T_h bijective and affine onto the reference triangle

$$T^r := conv\,\{(0,0),\ (1,0),\ (0,1)\}\ .$$

So for every $u \in V_h$ we have for every $(x,y) \in T^r$

$$\left(u \circ A_{T_h}^{-1}\right)(x,y) = u(a_{T_h}) + x\Big(u(b_{T_h}) - u(a_{T_h})\Big) + y\Big(u(c_{T_h}) - u(a_{T_h})\Big)$$

since the composition of linear functions is linear and therefore defined on T^r by the values at the vertices A_{T_h} is linear. We identify A_{T_h} with its Jacobian matrix. This gives on T_h with

$u = u \circ A_{T_h}^{-1} \circ A_{T_h}$

$$\nabla u = A_{T_h}^T \cdot \Big(u(b_{T_h}) - u(a_{T_h}) \, , \, u(c_{T_h}) - u(a_{T_h}) \Big)^T \quad \text{on } T_h$$

and so

$$\begin{aligned} \int\limits_{T_h} |\nabla u|^p dx \; &= \; |T_h| \cdot \Big| A_{T_h}^T \cdot \Big(u(b_{T_h}) - u(a_{T_h}) \, , \, u(c_{T_h}) - u(a_{T_h}) \Big)^T \Big|^p \\ &= \; \sqrt{P_{T_h}\Big(u(b_{T_h}) - u(a_{T_h}) \, , \, u(c_{T_h}) - u(a_{T_h}) \Big)}^{\,p} \, , \end{aligned}$$

where $P_{T_h} : \mathbb{R}^2 \to \mathbb{R}_{\geq 0}$ is a quadratic polynomial depending only on p and T_h. It holds

$$P_{T_h}(x,y) = 0 \quad \text{iff} \quad x = y = 0 \quad \text{for every } (x,y) \in \mathbb{R}^2 \, ,$$

since A_{T_h} is bijective and therefore A_{T_h} is regular. Therefore $u \mapsto \int\limits_{T_h} |\nabla u|^p dx$ is continuously differentiable on the open and dense set

$$D_{T_h}^1 := V_h \setminus \big\{ u \in V_h \;\big|\; u(a_{T_h}) = u(b_{T_h}) = u(c_{T_h}) \big\} \, .$$

For $u \in V_h$ and $p \geq 1$ we sum up

$$F_p(u) = \int\limits_{\overline{\Omega_h}} |\nabla u|^p dx = \sum_{T_h \in \tau_h} \sqrt{P_{T_h}\Big(u(b_{T_h}) - u(a_{T_h}) \, , \, u(c_{T_h}) - u(a_{T_h}) \Big)}^{\,p} \, . \tag{6.23}$$

For $p \geq 1$ we observe further that $F_p : V_h \to \mathbb{R}$ is continuously differentiable, hence strictly differentiable on the open and dense set

$$D^1 := \bigcap_{T_h \in \tau_h} D_{T_h}^1 \subseteq \Omega$$

, cf. Proposition 1.8.

Next we turn our attention to the function G_p for $p \geq 1$. The task to compute $G_p(v)$ for $v \in V_h$ and $p \geq 1$ is more challenging. Theoretically, it is possible to give a quadrature formula for $G_p(v)$ which is exact for every $p \geq 1$ and $v \in V_h$, but the evalutation of this formula is computation time consuming. Furthermore, to our knowledge such a formula is not implemented in FreeFEM++. Therefore we decided to approximate the function G_p. This is done by Gaussian quadrature.

For this purpose we define the nonlinear mapping

$$S_p : V_h \to V_h$$

such that for every $u \in V_h$ and every grid point $x_i \in \Omega$ holds

$$S_p(u)(x_i) = |u|^p(x_i) \ .$$

For every $T_h \in \tau_h$ as in (6.22) we choose q_{a,T_h}, q_{b,T_h}, $q_{c,T_h} > 0$ such that for every $v \in V_h$ holds

$$\int\limits_{T_h} v(x)dx \ = \ q_{a,T_h} v(a_{T_h}) + q_{b,T_h} v(b_{T_h}) + q_{c,T_h} v(c_{T_h}) \ ,$$

cf. [21, Section 5]. Thus for every $p \geq 1$ and $v \in V_h$ holds

$$\begin{aligned} \int\limits_{T_h} S_p(u)(x)dx \ &= \ q_{a,T_h} S_p(u)(a_{T_h}) + q_{b,T_h} S_p(u)(b_{T_h}) + q_{c,T_h} S_p(u)(c_{T_h}) \\ &= \ q_{a,T_h} |u|^p(a_{T_h}) + q_{b,T_h} |u|^p(b_{T_h}) + q_{c,T_h} |u|^p(c_{T_h}) \ . \end{aligned}$$

We observe that $u \mapsto \int\limits_{T_h} S_p(u)(x)dx$ is continuously differentiable on the open and dense set

$$D^2_{T_h} := V_h \setminus \left\{ u \in V_h \,\middle|\, u(a_{T_h}) = 0 \ \text{ or } \ u(b_{T_h}) = 0 \ \text{ or } \ u(c_{T_h}) = 0 \right\} \ .$$

Instead of G_p, we implemented the function $G^a_p : V_h \to \mathbb{R}$ which is given by

$$G^a_p(u) := G_1(S_p(u)) = \sum_{T_h \in \tau_h} \int\limits_{T_h} S_p(u)(x)dx = \sum_{i=1}^{dim\, V_h} q_i |u(x_i)|^p \ , \tag{6.24}$$

where the $q_i > 0$ are the coefficients of the exact quadrature formula for the grid points x_i of the mesh τ_h. Again for $p \geq 1$ the function $G^a_p : V_h \to \mathbb{R}$ is continuously differentiable on the open and dense subset of

$$D^2 := \bigcap_{T_h \in \tau_h} D^2_{T_h} \subset V_h \ .$$

Moreover for $p = 1$ we observe that for every $u \in V_h$ with $u \geq 0$ or $u \leq 0$ we have $|u| = S_1(u)$ and so $G_1(u) = G^a_1(u)$. (If u changes its sign in the interior of some $T_h \in V_h$, we have $|u| \neq S_1(u)$.) Taking into account that we are mainly interested in (6.12) and (6.15) and that some minimizer u of (6.12) and (6.15) is either non negative (or non positive) on the entirety of Ω we obtain that close to u we make little to no approximating error through the quadrature formula for G_1.

Later in Section 6.7.2 we observe that in the case $p = 10$ Algorithm 2.38 sometimes has problems approximating the minimizer of (6.9). We think that the error of the quadrature formula is one reason for those problems.

Now, we turn our attention to choosing and computing one element of $\partial F_p(u)$ and one element of $\partial G_p(u)$ for $u \in V_h$ and $p \geq 1$.

Lemma 6.25 *Let $p \geq 1$ and q solve $\frac{1}{p} + \frac{1}{q} = 1$ ($q = \infty$ if $p = 1$). Then $F_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ and $G_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ are locally Lipschitz continuous with respect to the Sobolev norm. For every $u \in W_0^{1,p}(\Omega)$ and every $u', v' \in W_0^{1,p}(\Omega)'$ holds $u' \in \partial F_p(u)$ and $v' \in \partial G_p(u)$ iff there exist $z_F \in L^q(\Omega, \mathbb{R}^n)$ and $z_G \in L^q(\Omega, \mathbb{R})$ with*

$$z_F(x) \in \Big(\partial |\cdot|^p\Big)\Big(\nabla u(x)\Big) \quad \text{and} \quad z_G(x) \in \Big(\partial |\cdot|^p\Big)\Big(u(x)\Big) \quad \text{for a.e. } x \in \Omega \tag{6.26}$$

such that for all $v \in W_0^{1,p}(\Omega)$ holds

$$\big\langle u' \,\big|\, v \big\rangle_{W_0^{1,p}(\Omega)} = \int_\Omega z_F(x) \cdot \nabla v(x) dx \quad \text{and} \quad \big\langle v' \,\big|\, v \big\rangle_{W_0^{1,p}(\Omega)} = \int_\Omega z_G(x) \cdot v(x) dx \,. \tag{6.27}$$

Proof. We recall that for a convex, locally Lipschitz continuous function $f : X \to \mathbb{R}$ the subdifferential of f at $x \in X$ and $\partial f(x)$ coincide, cf. Proposition 1.8. F_p and G_p are convex, locally Lipschitz continuous functions.
For $\tilde{n} \in \{1, n\}$ we define the convex, locally Lipschitz continuous function $\Psi_p : L^p(\Omega, \mathbb{R}^{\tilde{n}}) \to \mathbb{R}$

$$\Psi_p^{\tilde{n}}(w) \,=\, \int_\Omega |w(x)|^p dx$$

and the linear, bounded mappings $L_p : W_0^{1,p}(\Omega) \to L^p(\Omega, \mathbb{R}^n)$ and $Id_p : W_0^{1,p}(\Omega) \to L^p(\Omega, \mathbb{R})$ with $L_p u = \nabla u$ and $Id_p\, u = u$.
Then [13, Theorem 2.7.5] gives that for every $w \in L^p(\Omega, \mathbb{R}^{\tilde{n}})$ and every $w' \in L^p(\Omega, \mathbb{R}^{\tilde{n}})'$ holds $w' \in \partial \Psi_p^{\tilde{n}}(w)$ iff there exists $z_p^{\tilde{n}} \in L^q(\Omega, \mathbb{R}^{\tilde{n}})$ with

$$z_p^{\tilde{n}}(x) \in \Big(\partial |\cdot|^p\Big)\Big(w(x)\Big) \quad \text{for a.e. } x \in \Omega \tag{6.28}$$

such that for all $\tilde{w} \in L^p(\Omega, \mathbb{R}^{\tilde{n}})$

$$\big\langle w' \,\big|\, \tilde{w} \big\rangle_{L^p(\Omega, \mathbb{R}^{\tilde{n}})} = \int_\Omega z_p(x) \cdot \tilde{w}(x) dx \,. \tag{6.29}$$

Now [18, Proposition I.5.7] gives the claim for $F_p = \Psi_p^n \circ L_p$ with $z_F = z_p(\nabla u)$. Further, [18, Proposition I.5.7] gives the claim for $G_p = \Psi_p^1 \circ Id_p$ with $z_G = z_p^1(\nabla u)$. $\diamondsuit$

Corollary 6.30 *Let $p \geq 1$ and q solve $\frac{1}{p} + \frac{1}{q} = 1$ ($q = \infty$ if $p = 1$). Then $F_p : V_h \to \mathbb{R}$ and $G_p : V_h \to \mathbb{R}$ are locally Lipschitz continuous. For every $u \in V_h$ and every $u', v' \in V_h'$ holds $u' \in \partial F_p(u)$ and $v' \in \partial G_p(u)$ iff there exist $z_F \in L^q(\Omega, \mathbb{R}^n)$ and $z_G \in L^q(\Omega, \mathbb{R})$ with*

$$z_F(x) \in \Big(\partial |\cdot|^p\Big)\Big(\nabla u(x)\Big) \quad \text{and} \quad z_G(x) \in \Big(\partial |\cdot|^p\Big)\Big(u(x)\Big) \quad \text{for a.e. } x \in \Omega\,. \tag{6.31}$$

such that for all $v \in V_h$ holds

$$\big\langle u' \,\big|\, v \big\rangle_{W_0^{1,p}(\Omega)} = \int\limits_\Omega z_F(x) \cdot \nabla v(x) dx \quad \text{and} \quad \big\langle v' \,\big|\, v \big\rangle_{W_0^{1,p}(\Omega)} = \int\limits_\Omega z_G(x) \cdot v(x) dx \tag{6.32}$$

Proof. F_p is a convex, locally Lipschitz continuous function. Thus the subdifferential of F_p at $u \in X$ coincides with $\partial F_p(u)$ in both of the case $X = V_h$ and $X = W_0^{1,p}(\Omega)$, cf. Proposition 1.8. Assume some z_F satisfies (6.31) and (6.32). Then z_F as in Lemma 6.25 defines an elements u' of $W_0^{1,p}(\Omega)'$ through (6.27). By Lemma 6.25 it follows that u' is in the subdifferential of $F_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ at u. By the definition of the subdifferential, the functional u' restricted to V_h is in the subdifferential of $F_p : V_h \to \mathbb{R}$ at u.
Conversely, assume $u' \in \partial F_p(u) \subset V_h'$. Then, the generalized directional derivative $v \mapsto F_p^0(u; v)$ of $F_p : W_0^{1,p} \to \mathbb{R}$ is subadditive and positively 1-homogenous on the entirety of $W_0^{1,p}(\Omega)$ and $F_p^0(u; v) \geq \big\langle u' \,\big|\, v \big\rangle_{V_h}$ for all $v \in V_h$, cf. Proposition 1.3 and Definition 1.4. Note that (1.2) implies that the generalized directional derivative of $F_p : W_0^{1,p} \to \mathbb{R}$ is not smaller than the generalized directional derivative of $F_p : V_h \to \mathbb{R}$. The theorem of Hahn-Banach [59, Satz.III.1.3] gives the existence of some $\tilde{u}' \in W_0^{1,p}(\Omega)'$ with

$$\tilde{u}'\Big|_{V_h} = u' \quad \text{and} \quad F_p^0(u; v) \geq \big\langle \tilde{u}' \,\big|\, v \big\rangle_{W_0^{1,p}(\Omega)} \quad \text{for every } v \in W_0^{1,p}(\Omega)\,.$$

Thus $\tilde{u}' \in \partial F_p(u) \subset W_o^{1,p}(\Omega)'$. By Lemma 6.25 there exists some $z_F \in L^q(\Omega, \mathbb{R}^n)$ with (6.27). Thus the restriction (6.32) is satisfied too. The claim for G_p follows analoguosly. $\diamondsuit$

Thus in the implementation we have to choose the functions

$$z_F(u, x) = |\nabla u(x)|^{p-2} \nabla u(x) \quad \text{and} \quad z_G(u, x) = |u(x)|^{p-2} u(x) \quad \text{for } u \in V_h \text{ and } p \geq 1\,, \tag{6.33}$$

in the case that $p > 1$, where

$$z_F(u, x) := 0 \text{ if } \nabla u(x) = 0 \quad \text{and}$$

$$z_G(u,x) := 0 \text{ if } u(x) = 0 \,. \tag{6.34}$$

In the case $p = 1$ we have a choice if $u(x) = 0$ or $\nabla u(x) = 0$. In this case we choose z_F and z_G as in (6.33) and (6.34), too. Obviously these functions are measurable, since u and ∇u are measurable.
Thus, for every $u,\ v \in V_h$ and $p \geq 1$ the functions $\nabla u,\ z_F(u,\cdot),\ \nabla v$ and $z_F(u,\cdot)\cdot\nabla v$ are constant on every triangle $T_h \in \tau_h$. Therefore we easily find a quadrature formula which computes exactly $\int\limits_\Omega z_F(u,x)\cdot\nabla v(x)dx$ for every $u,\ v \in V_h$, cf. [21, Section 5].
We will not compute $\int\limits_\Omega z_G(u,x)\cdot v(x)dx$ for all $u,\ v \in V_h$ exactly in the implementation, since exact computations are too time consuming. Instead we will approximate this term. For this purpose we define the linear mapping

$$P :\ C(\Omega) \to V_h$$

such that for every $u \in C(\Omega)$ and every grid point $x_i \in \Omega$ holds

$$P\,u(x_i) = u(x_i) \,.$$

Let $u, v \in V_h$ and $p \geq 1$. With the exact (on V_h) quadrature formula from (6.24) we approximate:

$$\begin{aligned}
\int\limits_\Omega z_G(u,x)\cdot v(x)dx \;&\approx\; \int\limits_\Omega P\Big(z_G(u,\cdot)\cdot v\Big)(x)dx \\
&=\; \sum_{i=1}^{dim\,V_h} q_i P(z_G\Big(u,\cdot)\cdot v\Big)(x_i) \\
&=\; \sum_{i=1}^{dim\,V_h} q_i\, z_G(u,x_i)\cdot v(x_i) =: \big\langle u' \,\big|\, v\big\rangle_{V_h} \,,
\end{aligned} \tag{6.35}$$

where $u' \in V_h'$ is uniquely defined by (6.35). With this definition, (6.33) and (6.34) it holds

$$u' \in \partial G_p^a(u)$$

by (6.24), cf. Proposition 1.18.

We sum up. Since we can not efficiently compute $G_p(u)$ for $p \geq 1$ and $u \in V_h$ we replace $G_p(u)$ by $G_p^a(u)$. Instead of studying the restrictions of (6.12) (or (6.9)) and (6.15) to V_h, we

will study the following three minimization problems:

$$F_p(v) \to \min \quad \text{under } G_p^a(v) = 1 \quad \text{with } v \in V_h \tag{6.36}$$

$$E_p^a(v) := \frac{F_p}{G_p^a}(v) \to \min \quad \text{with } v \in V_h \quad \text{and} \tag{6.37}$$

$$\tilde{E}_p^a(v) := F_p(v) + K|1 - G_p^a|(v) \to \min \quad \text{with } v \in V_h \tag{6.38}$$

where $E_p^a : V_h \setminus \{0\} \to \mathbb{R}$, $\tilde{E}_p^a : V_h \to \mathbb{R}$ and $K > 0$ is sufficiently large. For (6.36), (6.37) and (6.38) we use exact quadrature formulae.

Lemma 6.39 *Let $K > \max\left\{F_p(u) \,\middle|\, u \in V_h : G_p(u) = 1\right\}$ and $p \geq 1$. The (6.36), (6.37) and (6.38) are equivalent problems.*
With equivalent we mean that for any $u \in V_h$ the following statements are equivalent:

1. *u is a minimizer of (6.36).*
2. *λu is a minimizer of (6.37) (for every $\lambda \in \mathbb{R} \setminus \{0\}$) and $G_p(u) = 1$.*
3. *u minimizes (6.38).*

Proof. We recall F_p and G_p^a are positively p-homogeneous. Therefore the fact that the problem (6.36) and the problem (6.37) are equivalent follows immediately.
Using the notation $\tilde{u} := \frac{u}{G_p^a(u)^{\frac{1}{p}}}$ for every $u \neq 0$ we have

$$\begin{aligned} E_p^a(\tilde{u}) = F_p(\tilde{u}) &= \left(F_p(u) - F_p(\frac{u}{G_p^a(u)^{\frac{1}{p}}}) G_p^a(u) \right) + F_p(\tilde{u}) \\ &= F_p(u) - F_p(\tilde{u})(G_p^a(u) - 1) \\ &\leq F_p(u) + K|G_p^a(u) - 1| = \tilde{E}_p^a(u) \,. \end{aligned}$$

In the case $G_p^a(u) \neq 1$ we have $E_p^a(\tilde{u}) < \tilde{E}_p^a(u)$ and in the case $G_p^a(u) = 1$ we have $E_p^a(\tilde{u}) = \tilde{E}_p^a(u)$. Thus, the minimizer $\bar{u} \in V_h$ of (6.38) satisfies $G_p^a(\bar{u}) = 1$ or $\bar{u} = 0$, what directly gives the claimed equivalence, since $u = 0$ is trivially no solution due to $\tilde{E}_p^a(0) = K$. ♢

Remark 6.40 In the application we do not have to take

$$K > \max\left\{F_p(u) \,\middle|\, u \in V_h : \; G_p(u) = 1\right\},$$

which looks at the first view very large and which is typically unkown, too. Typically, we use some $K > 0$ larger than some estimate of the first eigenvalue of the p-Laplace operator and

compute for (6.37) and (6.38) very similar results. E.g. we take $K := E_p^a(u_0)$ for some initial point u_0 or an estimate of the first eigenvalue is given from the literature.

The intersection of two open and dense sets is open and dense. The set

$$D^3 := \left\{ \tilde{u} \in V_h \,\middle|\, G_p^a(\tilde{u}) \neq 1 \right\}$$

is open and dense. Therefore $F_p : V_h \to \mathbb{R}$, $G_p^a : V_h \to \mathbb{R}$ and $|G_p^a - 1| : V_h \to \mathbb{R}$ are strictly differentiable on a common, open and dense set $D = D^1 \cap D^2 \cap D^3 \subset V_h$. Thus for $p \geq 1$ and every $K > 0$ the functions

$$E_p^a := \frac{F_p}{G_p^a} \quad \text{and} \quad \tilde{E}_p^a := F_p + K|G_p^a - 1|$$

are strictly differentiable on D too. Thus for every $u \in D$ Proposition 1.15 and Proposition 1.10 tell us how to compute the element of $\partial E_p^a(u)$ or the element of $\partial \tilde{E}_p^a(u)$, cf Proposition 1.8. In the unlikely case that $u \in V_h \setminus D$ we still formally apply the quotient (or sum) rule to compute some $u' \in V_h'$ and tacitly assume that $u' \in \partial E_p^a(u)$ (or that $u' \in \partial \tilde{E}_p^a(u)$), although the inclusion in Proposition 1.15 (or Proposition 1.10) might be strict.
In the following we omit the a in the notation to keep our notation simple, although e.g. G_p and G_p^a are not the same functions.

In practice we often have the situation that we know some $u' \in V_h'$ in the form that we can calculate for every $v \in V_h$ the term $\langle u' \,|\, v \rangle_{V_h}$, cf. (6.32), but we also need some representation of u' with respect to some given basis $\left\{ v_i' \,\middle|\, 1 \leq i \leq \dim V_h \right\}$ of V_h'. We now describe how we find this representation.
Let $\left\{ v_i' \,\middle|\, 1 \leq i \leq \dim V_h \right\}$ be a basis of V_h' and $\left\{ v_j \,\middle|\, 1 \leq j \leq \dim V_h \right\}$ be a basis of V_h. Further let $u' \in V_h'$ be fixed. To find a representation

$$\sum_{i=1}^{\dim V_n} a_i v_i' = u'$$

one has to solve the linear system of equations

$$\langle u' \,|\, v_j \rangle_{V_h} = \left\langle \sum_{i=1}^{\dim V_n} a_i v_i' \,\middle|\, v_j \right\rangle_{V_h} \quad \text{for } 1 \leq j \leq \dim V_h \,. \tag{6.41}$$

Several solvers for this equation are implemented in most FEM-boxes. We use the standard Conjugate Gradient Solver implemented in FreeFEM++. The theory of solving such problems

has been extensively studied in the literature and will not be investigated in this work. We refer to [30].

6.1.5 Difficulties of FEM Approaches

Although the minimizer of (6.12) can be approximated by the finite element method, up to our knowledge nobody has done it so far for $p < 1.07$. The reason for that is that it is already challenging to solve (6.36) for small dimensions of V_h. We deal with this difficulty now.

1. First of all, although F_p and G_p are convex for $p \geq 1$, the set $\left\{v \in BV(\Omega) \,\middle|\, G_p(v) = 1\right\}$ and the functions E_p and $\tilde{E}_p$ are not convex. Thus we can not use the well developed optimization theory for convex functions. We want to name here for example the Trust-Region Bundle method by Alt and Schramm, cf. [2, 52]. This theory has been partially generalized to locally Lipschitz continuous functions, cf. [52], but it is not clear if the additional assumptions in [52] are satisfied for the p-Laplace operator.

2. Moreover the dimension of V_h shouldn't be too small to gain reasonable approximations of the minimizer of (6.12), which means we can not take gradient sampling algorithms like in [4, 36], which have shown nice results for Hilbert spaces with low dimension. Only for spaces of the dimension smaller than 2 000 those algorithms work properly, which is by far too small to approximate properly the minimizer of (6.12). Typically V_h has dimension ranging from 2 000 up to 200 000 in this thesis.

3. Projected gradient methods, like the constrained descent method, don't give results for (6.12). In [28, 29], J. Horák used this method to find minimizer of (6.9) for $p \geq 1.1$. He didn't study $p < 1.1$, although the minimizers of (6.9) converge to some minimizer of (6.12) as $p \to 1$.

4. It also doesn't make sense to apply semismooth Newton like algorithms to (6.12) since e.g. the derivative of F_p is not semismooth.

5. To see why the problem (6.12) is so difficult, even if we restrain it to the finitely dimensional problem (6.36) we look at F_p. We recall that for $u \in V_h$ the function F_1 is given by (6.23), where the P_{T_h} are quadratic polynomials such that for every $(x, y) \in \mathbb{R}^2$ holds $P_{T_h}(x, y) = 0$ iff $(x, y) = (0, 0)$. It is well known that such functions are difficult to treat with standard gradient based methods. We recall that some minimizer u of (6.12) is (up to scaling) a characteristic function. Thus for an approximation $u^a \in V_h$ holds $u^a(b_{T_h}) - u^a(a_{T_h} \approx 0$ and $u^a(c_{T_h}) - u^a(a_{T_h}) \approx 0$ for most $T_h \in \tau_h$. Therefore the gradient of F_1 at u^a might not exist or oscillates strongly close to u^a where it exists. Typically also

the gradient of G_1 at u^a oscillates strongly close to u^a where it exists.
To visualize this effect we look at the gradient of E_1 close to a minimizer of (6.12). For the sake of simplicity we take $\Omega = (0,1)^2$. Then a good approximation $u^a \in V_h$ of a minimizer u of (6.12) is given in Figure 3. In Figure 4 we see a gradient of E_p at u^a which has lots

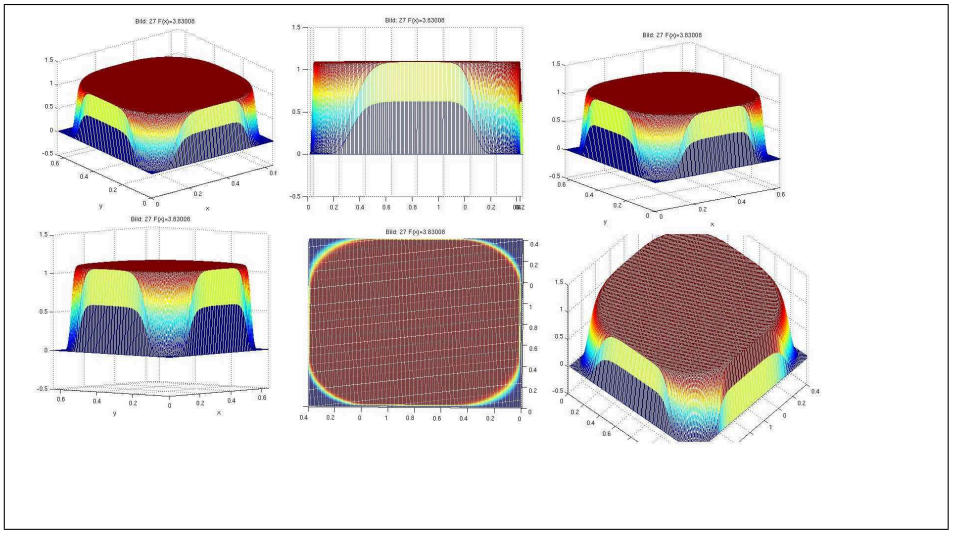

Figure 3: Different views of an approximation $u^a \in V_h$ of a minimizer $u = \frac{1}{|C|}\chi_C$ of (6.12) on the set $\Omega = (0,1)^2$, where C is the Cheeger set of Ω.

of peeks. Therefore it would lead to a descend only on a very small neighborhood. Due

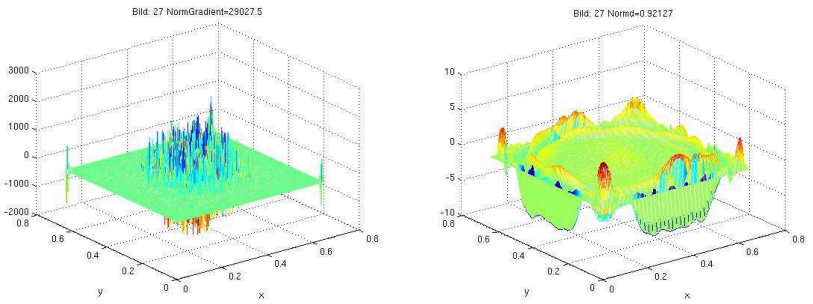

Figure 4: On the left hand side we see some gradient of E_p at u^a with u^a given in Figure 3 and on the right hand side we see an element of $\partial^\varepsilon E_p(u^a)$ for some small $\varepsilon > 0$, which was computed by Algorithm 3.3 after about 300 steps.

to rounding errors, classical steepest descend method fail to realize a descent in practice. In comparison in Figure 4 we see an element of $\partial^\varepsilon E_p(u^a)$ for some small $\varepsilon > 0$, which was computed by Algorithm 3.3 after about 300 steps. We can see that this descent direction is much better than the gradient of E_p at u^a, especially if we keep in mind that

we try to approximate (up to scaling) the characteristic function of the Cheeger set. This improvement of descent direction makes it possible to approach (6.12) numerically now, which wasn't possible before.

We point out that it does not help to smooth the functions F_1 and G_1, as it is common in many numerical applications, by taking for example

$$F_1^\varepsilon(v) := \int_\Omega \sqrt{(\nabla v)(x)^2 + \varepsilon^2}dx \quad \text{and} \quad G_1^\varepsilon(v) := \int_\Omega \sqrt{v(x)^2 + \varepsilon^2}dx.$$

The above problem remains. The gradients of $E_1^\varepsilon := \frac{F_1^\varepsilon}{G_1^\varepsilon}$, F_1^ε (and G_1^ε) at u^a oscillate dramatically for small ε too. Furthermore F_p and G_p with $p > 1$ are smooth approximations of F_1 and G_1. F_p and G_p show similar oscillations for $1 < p << 2$.

6.1.6 Calculating the Cheeger Set of Convex Set

Every characteristic function of a Cheeger set is (up to scaling) a minimizer of (6.12), cf. Section 6.1.2. For this purpose we calculate Cheeger sets in the case $\Omega \subset \mathbb{R}^2$ is open, bounded and convex with Lipschitz boundary. We recall that the Cheeger set is unique up to sets of Lebesgue measure 0. We make a similar ansatz to [16]. We recall a Cheeger set minimizes the quotient of perimeter and area of the set C, over all subsets C of Ω with $|C| \neq 0$. The Cheeger set of Ω is convex too. We restrict ourselves to sets $C \subseteq \Omega$ for which there exists a common point $x_0 \in \Omega$ and a function $r_C : [0, 2\pi] \to \mathbb{R}_{>0}$, such that a parametrization $\Phi_C : [0, 2\pi] \to \mathbb{R}^2$ of ∂C is given by

$$\Phi_C(\alpha) = x_0 + r_C(\alpha)(\cos\alpha, \sin\alpha) \quad \text{and}$$

$$C = \left\{ x_0 + r(\cos\alpha, \sin\alpha) \in \mathbb{R}^2 \;\middle|\; \alpha \in [0, 2\pi],\ r \in [0,\ r_C(\alpha)[\right\} \ . \tag{6.42}$$

Without loss of generality we assume $x_0 = 0$. Moreover we restrict ourselves to sets C such that Φ_C is given at finitely many points α_i and Φ_C is piecewise affine and continuous on the rest of $[0, 2\pi]$, which we concertize next.

For fixed $n \in \mathbb{N}$ we denote $\alpha_i = 2\pi\frac{i}{n}$ for $1 \leq i \leq n$. Furthermore we assign to every $y \in \mathbb{R}^n$ a parametrization of some set:

$$\Phi : \mathbb{R}^n \ \to\ C\left([0, 2\pi], \mathbb{R}^2\right)$$
$$(y_1,\, y_2, \ldots,\, y_n) = y \ \mapsto\ \Phi(y) \quad \text{with} \quad \Phi(y)(\alpha_i) = y_i(\cos\alpha_i, \sin\alpha_i)$$

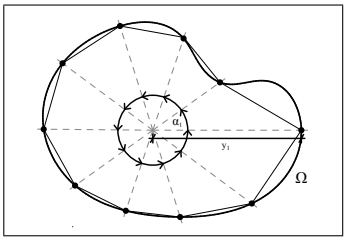

Figure 5: Computing the first eigenvalue for simple sets.

and $\Phi(y)$ is affine on $[\alpha_i, \alpha_{i+1}]$ for $0 \leq i \leq n-1$, where we set $y_0 := y_n$. Then the graph of $\Phi(y)$ is the boundary of

$$M(y) := \left\{ r(\cos\alpha, \sin\alpha) \,\middle|\, \alpha \in [0, 2\pi],\ 0 \leq r \leq |\Phi(y)(\alpha)| \right\} .$$

For $y \in \mathbb{R}^n$ we can calculate the area $A(M(y))$ of $M(y)$ by summing over the areas of the triangles

$$conv\left\{x_0,\ \Phi(y)(\alpha_i),\ \Phi(y)(\alpha_{i+1})\right\} \quad \text{with } \ 0 \leq i \leq n-1 \ .$$

Thus we obtain

$$\begin{aligned} A(M(y)) \ &:= \ \sum_{i=0}^{n-1} \frac{1}{2} \Big(\ \sin(\alpha_{i+1} - \alpha_i) \Big| \Phi(y)(\alpha_{i+1}) \Big| \Big) \Big| \Phi(y)(\alpha_i) \Big| \\ &= \ \frac{1}{2} \sin\frac{2\pi}{n} \sum_{i=0}^{n-1} y_i y_{i+1} \ . \end{aligned}$$

Using the invariance of the Euclidean norm under rotation we obtain for the permimeter:

$$\begin{aligned} P(M(y)) \ &:= \ \sum_{i=0}^{n-1} \Big| \Phi(y)(\alpha_{i+1}) - \Phi(y)(\alpha_i) \Big| \\ &= \ \sum_{i=0}^{n-1} \Big| y_{i+1} \left(\cos\alpha_{i+1}, \sin\alpha_{i+1}\right) - y_i \left(\cos\alpha_i, \sin\alpha_i\right) \Big| \\ &= \ \sum_{i=0}^{n-1} \left| y_{i+1} \left(\cos\frac{2\pi}{n}, \sin\frac{2\pi}{n} \right) - y_i \left(1, 0\right) \right| \\ &= \ \sum_{i=0}^{n-1} \sqrt{y_{i+1}^2 + y_i^2 - 2 y_i y_{i+1} \cos\frac{2\pi}{n}} \ . \end{aligned}$$

Now our goal will be to minimize

$$y \mapsto \frac{P(M(y))}{A(M(y))}$$

under the constraint

$$y_i(\cos\alpha_i, \sin\alpha_i) \in \Omega \quad \text{for } 0 \le i \le n \ .$$

There exists some $\phi_\Omega : [0, 2\pi] \to \mathbb{R}^2$ with $\phi_\Omega(\alpha) = r_\Omega(\alpha)(\cos\alpha, \sin\alpha)$ such that the graph of ϕ_Ω is the boundary of Ω. With this the constraint $y_i(\sin\alpha_i, \cos\alpha_i) \in \Omega$ becomes $0 \le y_i \le r_\Omega(\alpha_i)$, which can be realized in an easy way by adding the penalty function

$$(y_1, y_2, \dots, y_n) = y \mapsto \sum_{i=1}^{n} \max(0, -y_i) + \max\left(0, y_i - \phi_\Omega(\alpha_i)\right)$$

to the function $y \mapsto \frac{P(M(y))}{A(M(y))}$. The function $E_* : \mathbb{R}^n \setminus \{0\} \to \mathbb{R}$, which we actually minimize, is given by

$$E_*(y) := \frac{P(M(y))}{A(M(y))} + K \left(\sum_{i=1}^{n} \max(0, -y_i) + \max\left(0, y_i - \phi_\Omega(\alpha_i)\right) \right) , \tag{6.43}$$

where $K > 0$ is a sufficiently large constant. In practice we simply increase K if

$$\sum_{i=1}^{n} \max(0, -y_i) + \max\left(0, y_i - \phi_\Omega(\alpha_i)\right) \neq 0 \ .$$

Figure 6 shows the graph of $\Phi(y)$ evaluated at the minimizer y of (6.43), where Ω is a triangle. The minimizer was computed by Algorithm 2.38 in combination with the Algorithm 3.3 to gain the descent direction. Here we choose $n = 1000$. The calculation needs just a couple of minutes. In [16] it is not mentioned how long their algorithm took, so we can not compare the algorithms. The value of $E_*(u)$ at some minimizer $y \in \mathbb{R}^n$ of (6.43) is typically exact up to 8 digits if we compare tit to the analytically known value, cf. [32].

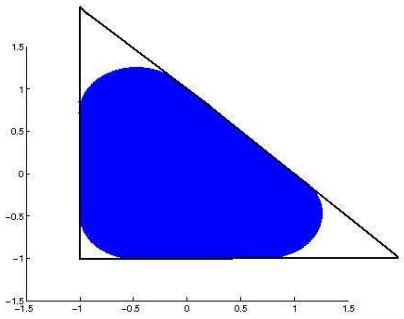

Figure 6: The Cheeger set of the triangle given by $conv\,\{(-1,1),\ (-1,2),\ (2,-1)\}$.

6.2 The Algorithm for Minimizing E_p

We recall that for every $u' \in V_h'$ and every $v \in V_h$ the expression $\langle u' \,|\, v \rangle_{V_h}$ is independent of the norm. But the predual $j(u')$ depends on the norm on V_h. Thus the descent direction d_k in Algorithm 2.38 depends on the norm. Therefore we discuss next the norm on V_h, before we discuss the Algorithm 2.38 in the framework of this section.

6.2.1 The Choice of the Norm

Now we discuss the norm on the subspace V_h. From the analytical point of view, two intuitive norms would be the usual $W^{1,p}$-norm

$$\|v\|_{W^{1,p}(\Omega)} := \left(\int_{\Omega_h} |v|^p + |\nabla v|^p dx \right)^{\frac{1}{p}} \tag{6.44}$$

and the $W_0^{1,p}$-norm

$$\|v\|_{W_0^{1,p}(\Omega)} := \left(\int_{\Omega_h} |\nabla v|^p dx \right)^{\frac{1}{p}} . \tag{6.45}$$

But these norms have two great disadvantages if $p \neq 2$. They will be outlined in the following. To determine the predual element $j(u')$ of some $u' \in V_h$, we have to solve the minimization problem

$$\left(v \mapsto \langle u' \,|\, v \rangle_{V_h} \right) \to \min \quad \text{under} \quad \|v\|_p = 1 \;\text{ with }\; v \in V_h \, . \tag{6.46}$$

But (6.46) is again a problem of dimension $\dim V_h$, which is comparably hard to solve as the problem (6.36), which we want to solve actually. In [29] the these norms have been used for the projected gradient method in the case $p \geq 1.1$. There was much computational time invested every time their algorithm had to compute the predual $j(u')$ of some $u' \in V_h'$.
We also tried (6.44) and (6.45) for $p \approx 1$ and Algorithm 2.38 in the beginning. But this ansatz turned out to be quite hopeless, because one has to solve (6.46) very accurately. Often we failed to compute the predual $j(u')$ sufficiently precise. Therefore we refrain to use (6.44) and (6.45) in this work. Also for $p > 1$ we will not use them. But later we will compare our results with the results in [29], cf. Section 6.7.1 and Section 6.7.2.
The second great disadvantage is that it is difficult to compute precisely a_{j+1} in Algorithm 3.34 for (6.44) and (6.45). We recall that we have to compute

$$a_{j+1} := \arg\min \left\{ \|c\| \;\middle|\; c \in conv\, C_j \right\} \, ,$$

where $C_j \subset V_h'$ is some finite set. Up to our experiences it is crucial to compute a_{j+1} very precise. Computing a_{j+1} becomes easy in the case that we consider a Hilbert space norm on V_h, because then we only have to minimize a quadratic function under linear constraints to get a_{j+1}, cf. Remark 3.5.

At the moment and for Algorithm 2.38 the advantages of a Hilbert space norm on V_h excel the advantages of the norms (6.44) and (6.45). For this reason we restrict ourselves to Hilbert space norms on V_h in this work. We leave the study of using other norms for Algorithm 2.38 to later work. Since $V_h \subset W_0^{1,2}(\Omega_h)$, there are at least 3 intuitive norms on V_h:

$$\|u\|_{L_2(\Omega)}^2 := \int_{\Omega_h} |u|^2(x)dx \ , \tag{6.47}$$

$$\|u\|_{W_0^{1,2(\Omega)}} := \|\nabla u\|_{L_2} \ , \tag{6.48}$$

$$\|u\|_{W^{1,2(\Omega)}} := \sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2} \ . \tag{6.49}$$

We will see that the right choice of norm to gain a fast convergence of the Algorithm 2.38 applied to (6.36), depends on the choice of $p \geq 1$. Up to our experience, in the case $p \geq 2$ the best norm is $\|\cdot\|_{W_0^{1,2}(\Omega)}$. For $1 \leq p < 1.5$ it is not so clear what the best choice of norm is, because it is not clear what is meant by "better". It appears to us that Algorithm 2.38 applied to $E_p(u_k)$ and $\tilde{E}_p(u_k)$ with $\|\cdot\|_{W_0^{1,2}(\Omega)}$ gives small values $E_p(u_k)$ and $\tilde{E}_p$ for smaller k than for the the other two norms. But close to the minimizer of (6.36) Algorithm 2.38 with $\|\cdot\|_{L_2(\Omega)}$ computes fastest very small values $E_p(u_k)$ and $\tilde{E}_p$. In Section 6.3.1 we will apply Algorithm 2.38 with all three Hilbert space norms to E_p and $\tilde{E}_p$ for $\Omega =]0,1[^2$ and compare the results.

Last we mention that in order to compute the derivative of F_p or G_p at $u \in V_h$ applied to a direction $v \in V_h$, we just have to evaluate the terms in (6.32) and (6.35) for the direction v. Thus we do not actually have to compute the gradient, which means that we do not have to solve the linear system of equations (6.41). This is crucial for the efficiency of Algorithm 3.9.

6.2.2 Common Settings

We recall that Algorithm 2.26 is a specialization of Algorithm 2.38 in the sense that $\|\cdot\|_k := \|\cdot\|$ and the model function $m_k := f$ for every $k \in \mathbb{N}$. In this chapter we will test Algorithm 2.38. In our computations we choose $\|\cdot\|_k := \|\cdot\|$ for $k \in \mathbb{N}$, where $\|\cdot\|$ is one of the three Hilbert space norms (6.47), (6.48) and (6.49). Moreover we choose $m_k := f := E_p$ or $m_k := f := \tilde{E}_p$ with

$p \geq 1$ for every $k \in \mathbb{N}$. Thus, we can apply Algorithm 3.3, which was formulated for Hilbert spaces, to Algorithm 2.38 (or Algorithm 2.26).
We recall that in in Algorithm 3.3 we compute a_{j+1} through (3.8), where λ^0 is the minimizer of the minimization problem (3.6) with respect to (3.7). We describe shortly how to compute λ^0 next. We again denote by $m \in \mathbb{N}$ the cardinality of C_j in Algorithm 3.3. It is well known that a minimizer $\lambda^0 \in \mathbb{R}^m$ of (3.6) with respect to (3.7) is characterized by the existence of Lagrange multipliers $\mu_l \geq 0$ $(1 \leq l \leq m)$ and $\nu \in \mathbb{R}$ which solve the Karush-Kuhn-Tucker equations, cf. [1, Satz 5.5.9] :

$$\begin{aligned} 0 &= A\lambda^0 + \nu e + \sum_{l=1}^{m} \mu_l e_l \\ 0 &= \sum_{l=1}^{m} \lambda_l^0 - 1 \\ 0 &= \lambda_l^0 \mu_l \text{ for } 1 \leq l \leq m, \end{aligned}$$

where $e_l \in \mathbb{R}^m$ is the l-th unit vector, $e := (1, \ldots, 1) \in \mathbb{R}^m$ and A is defined below (3.7).
The Fischer-Burmeister function $\Phi : \mathbb{R}^2 \to \mathbb{R}$ with

$$\Phi(x, y) := \sqrt{x^2 + y^2} - x - y \tag{6.50}$$

is semismooth, cf. [56, Prop. 2.26]. Moreover

$$\Phi(x, y) = 0 \quad \text{iff} \quad x \geq 0,\, y \geq 0 \;\text{ and }\; xy = 0 \qquad \text{for every } (x, y) \in \mathbb{R}^2 \,.$$

Thus, $\lambda, \mu \in \mathbb{R}^m$ and $\nu \in \mathbb{R}$ satisfy $\lambda \geq 0$, $\mu \geq 0$ and the Karush-Kuhn-Tucker equations if and only if they are some zero of the function $F : \mathbb{R}^{2m+1} \to \mathbb{R}^{2m+1}$ with

$$F_{Ne}(\lambda, \nu, \mu) := \begin{pmatrix} A\lambda + \nu e + \sum_{l=1}^{m} \mu_l e_l \\ \sum_{l=1}^{m} \lambda_l - 1 \\ \Phi(\lambda_1, \mu_1) \\ \Phi(\lambda_2, \mu_2) \\ \vdots \\ \Phi(\lambda_m, \mu_m) \end{pmatrix} .$$

The function F_{Ne} is semismooth by Proposition 4.12. Thus, we can apply the exact semismooth Newton method to F_{Ne}, cf. Remark 4.16. In this chapter we will always compute a_{j+1} in Al-

gorithm 3.3 by (3.8), where λ^0 is some zero of F_{Ne}, which has been computed with the exact semismooth Newton method.

For easier references we formulate the Algorithm 2.38 with all special parameters and used algorithm as one algorithm next.

Algorithm 6.51 *Let $\|\cdot\|$ be one of the Hilbert space norms (6.47), (6.48) and (6.49) and f be the energy function F_p or $\tilde{E}_p$. Apply Algorithm 2.38 to f with the following specialization:*

- $\|\cdot\|_k := \|\cdot\|$ *and* $m_k := f$ *for every* $k \in \mathbb{N}$,
- *use Algorithm 3.3 to compute $D_{k,i}$ in Step 4 of Algorithm 2.38 (or Step 2 of Algorithm 2.26),*
- $$\begin{aligned} \delta' = 0.3,\ \delta = 0.2,\ \varepsilon_{0,0} = 0.08 \quad \textit{and} \\ g(x,y) = y,\ H(x) = \frac{x}{2},\ h(x) = \frac{6x}{\varepsilon_{0,0}}\ , \end{aligned} \tag{6.52}$$
- *for every* $j \in \mathbb{N}$
$$C_j := \{a_0,\ a_j\} \cup \left\{b_l \;\middle|\; \max\{0,\ j-8\} < l \leq j\right\}$$
in Algorithm 3.3,
- *a_{j+1} in Algorithm 3.3 is computed through (3.8), where λ^0 is a zero of F_{Ne}, which we compute with the exact semismooth Newton method,*
- *in every descent step use the following Algorithm 6.53 to compute σ_k and $x_{k+1} := x_k - \sigma_k d_k$ and*
- *choose $\varepsilon_{k,i+1} := H(\varepsilon_{k,i})$ and $\varepsilon_{k+1,0} := \varepsilon_k$ for every $k, i \in \mathbb{N}$.*

Figure 7 shows a flow diagram of Algorithm 6.51. For later references we allow that the norm might be changed after every descent step in Figure 7, although we will not do this in Algorithm 6.51. But to keep the notation simple, we refrain to consider a change of the model function in every iteration step.

We can see that the computation time consuming parts of Algorithm 6.51 are:

1. Computing an element of $\partial E_p(u)$ or $\partial \tilde{E}_p(u)$ for some $u \in V_h$, i.e. solving (6.41),
2. a line search to make a descent step and
3. calling the Algorithm 3.9.

The computation time of solving (6.41) and of the line search of course depend on the desired accuracy. We discuss these points next.

1. When we compute an element u' of $\partial E_p(u)$ or $\partial \tilde{E}_p(u)$ for some point $u \in V_h$, thus solving (6.41), we take just the standard setting for the "solve" command in FreeFEM++ in the case $1 \leq p \leq 2$. For large p we have to compute u' more precise in (6.41) or the Algorithm 6.51 gets stuck.
2. We use the following line search algorithm.

Algorithm 6.53 (Line Search)
Let the functions u_k, $D_k \in V_h$, the norm $\|\cdot\|$ and $\delta, \varepsilon_k > 0$ be given. Let $f := E_p$ or $f := \tilde{E}_p$. We compute σ_k next.

(a) Set $\sigma_k = \varepsilon_k$ and $\tilde{\sigma} = 1000\varepsilon_k$.

(b) Set $l = 1$.

(c) Set $m = 1$.

(d) If

$$f\Big(u_k - \tilde{\sigma}\frac{D_k}{\|D_k\|}\Big) - f(u_k) < -\delta\,\|D_k\|\,\tilde{\sigma} \quad \textit{and} \quad m < 10$$

set

$$\sigma_k = \sigma_k + \tilde{\sigma}\ ,$$

increment m by 1 and go to Step (d).
Else set

$$\tilde{\sigma} = 0.1\tilde{\sigma}$$

and increment l by 1.
If $l = 9$ stop Algorithm 6.53 and return σ_k, else go to Step (c)

Notice that in the end $\tilde{\sigma} = \varepsilon_k \cdot 10^{-5}$. It doesn't make sense to compute with smaller $\tilde{\sigma}$ since in the next step we try to find a descent direction on $\overline{B_{V_h}(u_{k+1}\,,\,\varepsilon_k)}$ anyway. Thus we compute elements of $\partial^{\varepsilon_k} E_p(x_{k+1})$ or $\partial^{\varepsilon_k} \tilde{E}_p(x_{k+1})$.

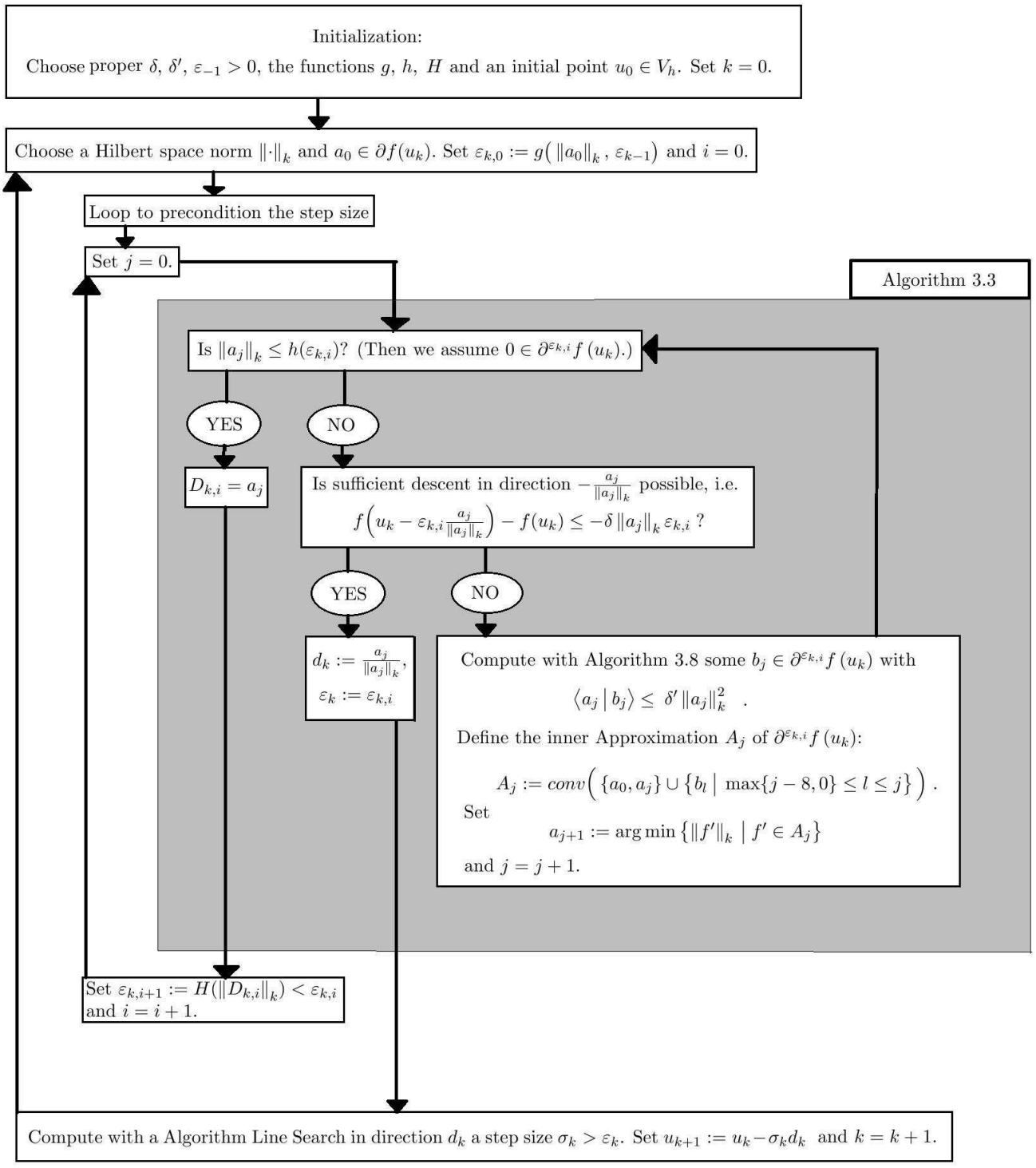

Figure 7: A flow diagram of Algorithm 6.51

3. For $p \leq 2$ the Algorithm 3.9 is usually fast and reliable, even so the functions E_p and $\tilde{E}_p$ are not smooth. But occasionally it happened in the beginning of our research that if $\|u_k\|_p << 1$, Algorithm 3.9 did not abort for E_p. Notice that E_p is highly oscillating close to $0 \in V_h$ since E_p is positively 1-homogeneous. Thus small rounding and quadrature errors might have a large effect. For this reason, we usually replaced u_k by $u_k \cdot \frac{10}{\|u_k\|p}$ after incrementing k by one in the case that we used E_p. Since E_p is positively 1-homogeneous, this doesn't change the value of E_p, but Algorithm 3.9 works fine. For $\tilde{E}_p$ no such problems occurred.

The by far most time consuming part of Algorithm 6.51 is computing a gradient, in particular if the dimension of the space is large. Therefore we count the gradients calculations, to compare the speed of the Algorithms 6.51 for the different settings. We also count the number of steps of Algorithm 6.51, were we count each null step and each descent step as **one step**. Recall Algorithm 2.38/ Algorithm 6.51 makes a null steps in the case that the null step assumption (NSAs) is satisfied and a descent steps else. Thus, in the case that Algorithm 6.51 makes a descent step the descent step assumption (DSAs) is satisfied. Observe that the number of steps is equal to the number of calls of Algorithm 3.3. We avoid talking of iterations, because it might be confusing, which iteration of Algorithm 6.51 we mean. Notice that Algorithm 6.51 is formulated with two for loops.

6.3 The 1-Laplace operator

6.3.1 The 1-Laplace operator on the Square

Now we compare the convergence of Algorithm 6.51 applied to the functions E_p and $\tilde{E}_p$. Since we will study the p-Laplace operator also on the square later, we first formulate the problem for general $p \geq 1$.
We will vary the norm on V_h and compare the convergence of Algorithm 6.51 when we use the different Hilbert space norms (6.47), (6.48) and (6.49). For this purpose we choose $\Omega = (0,1)^2$ fix and create fixed meshes for different numbers of grid points. We define an equidistant mesh on $\overline{\Omega} = [0,1]^2$ by using the command "mesh Omega=square(n,n);" in FreeFEM++, where dim $V_h = (n-2)^2$ and $n \in \{41,\ 81,\ 161\}$. On this mesh we consider the space V_h of all continuous function which are affine on every triangle of the mesh. This is also implemented in FreeFEM++ by using the command "fespace Vh(Omega,P1)".

As starting points we use the following functions u_0^1, $u_0^2\, u_0^3\, u_0^4 \in V_h$, which are uniquely defined through their values at every grid point $(x_i, y_i) \in \overline{\Omega}$ with $1 \leq i \leq (\dim V_h + 2)^2$:

$$u_0^1(x_i, y_i) \ = \ \sin(x_i \pi) \sin(y_i \pi) \ ,$$

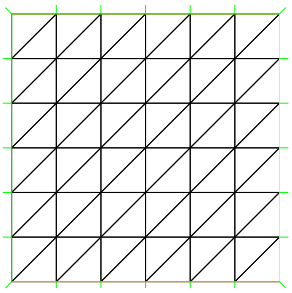

Figure 8: The mesh created by "mesh Omega=square(n,n);"

$$\begin{aligned} u_0^2(x_i, y_i) &= 1.2\chi_{[.1,.9]^2}(x_i, y_i)\ , \\ u_0^3(x_i, y_i) &= 1.2\chi_{[.05,.95]^2}(x_i, y_i) \quad \text{and} \\ u_0^4(x_i, y_i) &= 1.2\chi_{[.01,.99]^2}(x_i, y_i)\ . \end{aligned}$$

In the case $n = 80$ we also use $u_0^5 \in V_h$ defined through

$$u_0^5(x_i, y_i) = u_{500}^s(x_i, y_i)$$

as starting point, where u_{500}^s is the result of Algorithm 6.51 after 500 steps, applied to the norm $\|\nabla u\|_{L^2(\Omega)}$ and $\tilde{E}_p$ with $K = 50$ and initial point u_0^3 for $n = 41$.

The Speed of the Algorithm in the Case p=1 :

Now we study the case $p = 1$. We recall that a minimizer of (6.12) is given by $\frac{1}{|C|}\chi_C$, where C is the open Cheeger set of $\Omega =]0,1[^2$, cf. Figure 2. Thanks to [32] for a simple Ω like $(0,1)^2$ it is an easy task to determine the first eigenvalue of the 1-Laplace operator analytically by using the observation that the curvature of the boundary of the Cheeger set is constant in the interior of Ω and that for convex Ω the Cheeger set is convex and unique. Here we just give the result and refer to [32] for more details: The first eigenvalue λ_1 for $\Omega = (0,1)^2$ is given by

$$\lambda_1 = E_p\Big(\frac{1}{|C|}\chi_C\Big) = \tilde{E}_p\Big(\frac{1}{|C|}\chi_C\Big) = (2 + \sqrt{\pi}) \approx\ 3.7725\ . \tag{6.54}$$

This means that if we use $\tilde{E}_p$ we have to choose at least $K > 3.7725$. We choose

$$K = 5,\ K = 50 \quad \text{and} \quad K = 500\ .$$

To gain an upper bound for the minimal value of E_1 and $\tilde{E}_1$ on V_h we define the function $\chi_C^a \in V_h \subseteq W_0^{1,1}(\Omega)$ with

$$\chi_C^a(x_i) = \chi_C(x_i) \quad \text{on every grid point } (x_i) \in \overline{\Omega}\,. \tag{6.55}$$

Note that we assume C to be open, so this functions is well defined. Thus

$$\min_{u\in V_h} E_1(u) \le F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big) \quad \text{and} \quad \min_{u\in V_h} \tilde{E}_1(u) \le F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)\,.$$

$F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$ gives us further a first impression of the discretization error. We compute:

$n =$	41	81	161
$F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big) =$	4.10954	4.0783	4.06823

We recall that the quadrature formulae, which we use, are exact for $G_1(\chi_C^a)$ and $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$.

In the following we compare Algorithm 6.51 with the steepest descent method (SDM) and the projected gradient method (PGM). All three algorithms have been implemented by ourselves. Therefore we do not claim that these implementations are the state of the art. But they all use the same source code to evaluate terms like $E_p(u)$ or an element of $\partial E_p(u)$. Also the subroutines like e.g. line search are the same. This way we hope to keep the algorithms comparable.
SDM and PDM are designed to find minimizer of smooth functions. E_p and $\tilde{E}_p$ are not smooth. We apply these algorithms anyway, to see how far they proceed. We haven't implemented any other nonsmooth algorithms to find a minimizer of (6.12) yet; as nobody else has done this up to our knowledge. PGM has been used by J. Horák in [29] for $p \ge 1.1$ with the norms $\|\cdot\|_{W_0^{1,p}(\Omega)}$ and $\|\cdot\|_{W^{1,p}(\Omega)}$. That is also the main reason, why we use PGM here. But we can not apply PGM with these norms for $p = 1$ too. In order to compute a projected gradient of F_p, J. Horák used an augmented Lagrangian method, which only works for smooth functions. Therefore we use the Hilbert space norms (6.47), (6.48) and (6.49) for PGM in the case $p = 1$.

The next table shows the final function values of F_1 of the PGM and the final function values of E_1 and $\tilde{E}_1$ of the SDM. It also shows the number of used gradients of the SDM and PGM with the different norms. The mesh had 41^2 grid points and the initial function was u_0^1. The SDM stopped in the case that $E_1(u_k) = E_1(u_{k-1})$ (or $\tilde{E}_1(u_k) = \tilde{E}_1(u_{k-1})$) and PGM stopped in the case $F_1(u_k) = F_1(u_{k-1})$.

Steepest Descent (SDM) and Projected Gradient Method (PGM)				
$p = 1$	$n = 41$			Norm: $\|u\|_{L_2}$
The results for SDM:				
Function	$\frac{F_p}{G_p}$	$F_p + 5\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	5.239	5.096	4.889	4.776
Gradients	2500	2500	30000	30500
The PGM stops after 3020 gradient calculations with the value 5.234.				
$p = 1$	$n = 41$			Norm: $\|\nabla u\|_{L_2}$
The results for SDM:				
Function	$\frac{F_p}{G_p}$	$F_p + 5\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	4.083	4.159	5.046	5.082
Gradients	446	173	8	7
The PGM stops after 1002 gradient calculations with the value 4.15.				
$p = 1$	$n = 41$			Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$
The results for SDM:				
Function	$\frac{F_p}{G_p}$	$F_p + 5\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	4.191	4.138	5.038	5.075
Gradients	831	191	6	7
The PGM stops after 714 gradient calculations with the value 4.19.				

One can see that the minimal values computed by SDM for E_1 and $\tilde{E}_1$ and the minimal values computed by PGM for F_1 are not even close to the minimal values of (6.37), (6.38) and (6.36). If we use the norm $\|u\|_{L_2}$ the method is very slow right from the start. Figure 9 shows results for the norm $\|\nabla u\|_{L^2(\Omega)}$. The left hand side shows a result of SDM and the center shows the result for PGM.

In [29] no results for $p < 1.1$ were achieved.

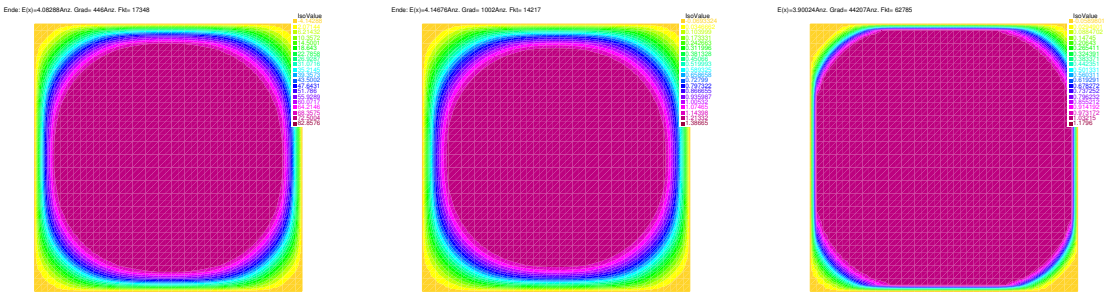

Figure 9: The left hand side shows the level sets of the result of SDM for E_p with the norm $\|\nabla u\|_{L^2(\Omega)}$. In the center we see the level sets of the result of PGM with the norm $\|\nabla u\|_{L^2(\Omega)}$. The right hand side shows the result of Algorithm 6.51 for Ep and the norm $\|\nabla u\|_{L^2(\Omega)}$.

Next we turn to Algorithm 6.51. The following tables show, depending on the choice of function and norm, which value the function at the point u_k in the $k-$th step has and how many calculations of gradients were necessary. Typically the algorithm was stopped in the case that $k = 501$. We programmed carefully and implemented several tests. When computing b_j in Algorithm 3.3 we had the following test. We recall Algorithm 3.9 formally computes some b_j which satisfies

$$\langle b_j \,|\, a_j \rangle \leq \delta_1 \|a_j\|^2 \ . \tag{6.56}$$

Then the FreeFEM++ routine "solve" solves (6.41) to get a representation of b_j with respect to some basis. We tested if (6.56) still holds for the representation of b_j. If not we stopped the algorithm. This is the reason, why some computations are stopped earlier.

F

6.3.2 Norm: $\|\nabla u\|_{L_2}$

Algorithm 6.51								
$p = 1.0$		E_1				Norm: $\|\nabla u\|_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$E_p(u_k)$ =	5.255	5.1016	4.537	4.0850	5.252	4.043	3.895
10	$E_p(u_k)$ =	4.3730	4.3692	4.2848	4.0488	4.6804	4.0163	3.8949
	Gradients	10	20	30	48	10	77	29
25	$E_p(u_k)$ =	4.2225	4.2517	4.1737	3.9968	4.1513	3.9575	3.8948
	Gradients	25	35	45	64	25	92	47
50	$E_p(u_k)$ =	4.2075	4.2496	4.1535	3.9951	4.1375	3.9550	3.8948
	Gradients	60	74	88	124	60	150	86
75	$E_p(u_k)$ =	4.2073	4.2486	4.1517	3.9913	4.1375	3.9509	3.8948
	Gradients	88	134	166	203	88	224	144
100	$E_p(u_k)$ =	4.2070	4.2462	4.1484	3.9861	4.1373	3.9430	3.8948
	Gradients	113	198	259	338	113	321	221
150	$E_p(u_k)$ =	4.2049	4.2306	4.1362	3.9669	4.1360	3.9254	3.8948
	Gradients	163	338	527	726	163	727	393
200	$E_p(u_k)$ =	4.2000	4.1690	4.0947	3.9195	4.1305	3.8890	3.8947
	Gradients	250	531	900	1358	229	1323	614
250	$E_p(u_k)$	4.1950	3.9773	3.9304	3.9022	4.1268	3.8480	3.8942
	Gradients	391	838	1423	2587	362	2374	850
300	$E_p(u_k)$	4.1686	3.9048	3.9011	3.9002	4.1114	3.8404	3.8924
	Gradients	559	1739	3325	5718	525	4591	1175
350	$E_p(u_k)$	4.0686	3.9001	3.8998	3.8998	4.0552	3.8387	3.8881
	Gradients	774	5572	12214	14416	763	9830	1643
400	$E_p(u_k)$	3.9305	3.8997	3.8997	3.8997	3.9274	3.8382	3.8831
	Gradients	1324	16705	24070	26240	1223	18325	2507
450	$E_p(u_k)$	3.9015	3.8997	3.8997	3.8997	3.8630	3.8381	3.8784
	Gradients	2926	28458	36970	39461	2305	27841	3778
500	$E_p(u_k)$	3.8999	3.8997	3.8997	3.8997	3.8444	3.8380	3.8756
	Gradients	9751	39422	48327	51538	3896	38103	5163

$*$: Computer shutdown before the end of the computation.

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 5\lvert G_1 - 1\rvert$				Norm: $\lVert\nabla u\rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	5.103	5.074	4.572	5.3767	5.102	5.585	3.895
10	$\tilde{E}_1(u_k)$ =	4.7091	4.8677	4.5133	4.5427	4.7301	4.8800	3.8950
	Gradients	10	23	24	13	10	16	33
25	$\tilde{E}_1(u_k)$ =	4.4098	4.7967	4.4976	4.0991	4.3574	4.2813	3.8944
	Gradients	30	40	52	36	30	35	124
50	$\tilde{E}_1(u_k)$ =	4.3895	4.7915	4.4929	4.0883	4.3209	4.0526	3.8921
	Gradients	70	138	150	124	70	99	312
75	$\tilde{E}_1(u_k)$ =	4.3464	4.7832	4.4882	4.0843	4.2011	4.0419	3.8889
	Gradients	113	245	293	280	120	290	617
100	$\tilde{E}_1(u_k)$ =	4.2778	4.7694	4.4802	4.0782	4.1025	4.0311	3.8856
	Gradients	158	362	457	506	228	691	1078
150	$\tilde{E}_1(u_k)$ =	4.0718	4.6480	4.4019	4.0321	3.9046	3.9036	3.8815
	Gradients	389	647	865	1048	872	1563	2431
200	$\tilde{E}_1(u_k)$ =	3.9408	4.0058	4.0819	3.9245	3.8574	3.8445	3.8787
	Gradients	952	968	1335	1756	2439	2978	4066
250	$\tilde{E}_1(u_k)$ =	3.9018	2325	3.9061	3.9016	3.8408	3.8383	3.8772
	Gradients	2606	3.9049	2538	3969	4569	11300	5834
300	$\tilde{E}_1(u_k)$ =	3.9000	3.9001	3.9001	3.9000	3.8384	3.8382	3.8761
	Gradients	8385	8345	9446	10607	13068	26136	7773
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8998	3.8997	3.8998	3.8383	3.8381	3.8750
	Gradients	19322	14502	20755	21156	27218	42321	9782
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8383	3.8381	3.8738
	Gradients	32845	33127	34246	33914	44887	58068	11970
450	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8996	3.8997	3.8383	3.8381	3.8727
	Gradients	46435	48038	49827	46548	61984	61499	14210
500	$\tilde{E}_1(u_k)$ =	3.8997	$*$	$*$	3.8997	3.8383	3.8381	3.8716
	Gradients	53658	$*$	$*$	61127	72735	63927	16419

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 50\lvert G_1 - 1\rvert$				Norm: $\lVert\nabla u\rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	31.88	17.18	7.939	11.77	31.87	13.26	3.910
10	$\tilde{E}_1(u_k)$ =	4.6484	5.0298	4.5272	4.1837	4.8526	4.5826	3.8962
	Gradients	18	16	19	18	16	19	20
25	$\tilde{E}_1(u_k)$ =	4.3131	4.8485	4.5189	4.0924	4.3136	4.0692	3.8942
	Gradients	48	46	53	50	46	55	84
50	$\tilde{E}_1(u_k)$ =	4.2735	4.8368	4.5096	4.0856	4.2481	4.0562	3.8927
	Gradients	118	127	136	151	119	163	269
75	$\tilde{E}_1(u_k)$ =	4.2471	4.8235	4.4986	4.0819	4.1968	4.0488	3.8893
	Gradients	218	232	251	324	222	369	495
100	$\tilde{E}_1(u_k)$ =	4.2122	4.8024	4.4854	4.0783	4.1047	4.0315	3.8851
	Gradients	315	358	402	518	354	605	864
150	$\tilde{E}_1(u_k)$ =	4.0709	4.6499	4.4082	4.0411	3.9157	3.9080	3.8795
	Gradients	585	656	763	910	808	1177	2182
200	$\tilde{E}_1(u_k)$ =	3.9438	4.0000	4.1783	3.9200	3.8427	3.8396	3.8763
	Gradients	1078	983	1139	1426	2542	3966	3781
250	$\tilde{E}_1(u_k)$ =	3.9012	3.9025	3.9052	3.9008	3.8382	3.8381	3.8734
	Gradients	2994	2291	2096	3808	12366	17015	5711
300	$\tilde{E}_1(u_k)$ =	3.8998	3.8999	3.8999	3.8998	3.8380	3.8380	3.8705
	Gradients	11745	11840	9408	13061	26138	29772	7433
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8380	3.8677
	Gradients	23966	25011	22169	24868	42491	43770	9307
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8380	3.8651
	Gradients	39121	38773	35324	39564	55691	57163	11333
450	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8996	3.8997	3.8380	3.8380	3.8623
	Gradients	53259	56259	47630	53504	69460	68885	13484
500	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8996	3.8997	3.8380	3.8380	3.8597
	Gradients	67439	71565	61713	67352	81692	82793	15800

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 500\lvert G_1 - 1\rvert$				Norm: $\lVert\nabla u\rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	299.7	138.2	41.62	75.75	299.5	90.00	4.052
10	$\tilde{E}_1(u_k)$ =	4.6314	5.1162	4.5422	4.2778	4.4827	4.7574	3.9094
	Gradients	18	16	19	18	16	19	20
25	$\tilde{E}_1(u_k)$ =	4.4271	4.8647	4.5200	4.1752	4.3714	4.2380	3.9019
	Gradients	48	46	51	48	46	55	61
50	$\tilde{E}_1(u_k)$ =	4.3739	4.8396	4.5107	4.1465	4.2755	4.2203	3.8936
	Gradients	106	126	136	109	101	162	178
75	$\tilde{E}_1(u_k)$ =	4.3049	4.8177	4.4986	4.0833	4.2226	4.2034	3.8888
	Gradients	170	228	261	184	194	316	390
100	$\tilde{E}_1(u_k)$ =	4.2495	4.7976	4.4859	4.0778	4.1280	4.1135	3.8850
	Gradients	262	356	442	361	324	466	788
150	$\tilde{E}_1(u_k)$ =	4.1021	4.6718	4.3992	4.0472	3.9067	3.9036	3.8801
	Gradients	510	652	894	769	907	1011	2011
200	$E_p(u_k)$ =	3.9407	4.0283	4.1151	3.9307	3.8523	3.8396	3.8776
	Gradients	1041	1020	1421	1258	2479	3818	3624
250	$E_p(u_k)$ =	3.9012	3.9021	3.9028	3.9009	3.8390	3.8380	3.8755
	Gradients	2831	2579	2837	3293	6662	17185	5289
300	$E_p(u_k)$ =	3.8998	3.8998	3.8998	3.8998	3.8381	3.8380	3.8734
	Gradients	13767	11837	11865	13296	20522	29319	7090
350	$E_p(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8381	3.8380	3.8713
	Gradients	25936	24830	25064	27231	35400	44390	8916
400	$E_p(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8381	3.8380	3.8692
	Gradients	41571	37019	39570	39991	49542	57106	10830
450	$E_p(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8381	3.8380	3.8672
	Gradients	56434	48688	54555	53754	64440	70412	12876
500	$E_p(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8381	3.8380	3.8653
	Gradients	69855	59885	67204	65717	79051	80277	15002

6.3.3 L_2 Norm

Algorithm 6.51								
$p = 1.0$		E_1				Norm: $\|u\|_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$E_p(u_k)$ =	5.2546	4.102	4.537	4.085	5.252	4.0426	3.895
10	$E_p(u_k)$ =	4.9782	4.9033	4.4341	4.0362	4.9811	4.0426	3.8887
	Gradients	64	225	222	727	215	4263	2003
25	$E_p(u_k)$ =	4.9703	4.8962	4.4312	3.9798	4.9646	4.0426	3.8880
	Gradients	81	242	240	1112	235	11808	2070
50	$E_p(u_k)$ =	4.9569	4.8886	4.4281	3.9343	4.9575	4.0425	3.8874
	Gradients	112	271	280	1901	289	12820	2288
75	$E_p(u_k)$ =	4.9297	4.8810	4.4244	3.9144	4.9444	4.0425	3.8867
	Gradients	159	322	356	3648	372	12846	2588
100	$E_p(u_k)$ =	4.9057	4.8686	4.4203	3.9058	4.9130	4.0425	3.8856
	Gradients	220	408	452	6554	493	12907	3008
150	$E_p(u_k)$ =	4.7494	4.8083	4.3914	3.9010	4.7201	4.0424	3.8838
	Gradients	442	691	784	15136	894	13266	4211
200	$E_p(u_k)$ =	4.2437	4.6029	4.2890	3.9001	4.3167	4.0423	3.8816
	Gradients	894	1176	1352	25312	1606	13968	5877
250	$E_p(u_k)$ =	3.9556	4.1514	4.0314	3.8999	3.9689	4.0407	3.8815
	Gradients	2055	1848	2184	34630	2870	14907	7831
300	$E_p(u_k)$ =	3.9094	3.9235	3.9192	3.8998	3.8742	4.0272	3.8805
	Gradients	4993	3334	4246	45018	5650	15863	9804
350	$E_p(u_k)$ =	3.9012	3.9051	3.9027	3.8998	3.8491	3.9859	3.8795
	Gradients	13809	8832	10503	55834	12708	16913	11916
400	$E_p(u_k)$ =	3.8997	3.9002	3.9002	38998	3.8418	3.9338	3.8785
	Gradients	21974	19768	20942	67312	22010	18255	13988
450	$E_p(u_k)$ =	3.8997	3.8998	3.8998	3.8998	3.8392	3.8940	3.8775
	Gradients	32786	30111	32207	78552	32140	20433	16201
500	$E_p(u_k)$ =	3.8997	3.8997	3.8998	3.8998	3.8385	3.8714	3.8765
	Gradients	44511	41196	41913	89974	42423	24335	18468

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 5\lvert G_1 - 1\rvert$				Norm: $\lVert u \rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	5.1031	5.074	4.572	5.377	5.102	5.585	3.895
10	$\tilde{E}_1(u_k)$ =	4.6439	4.7288	4.3966	4.2083	4.8430	4.7837	3.8915
	Gradients	337	343	385	113	821	1114	4152
25	$\tilde{E}_1(u_k)$ =	4.2839	4.3257	4.1958	4.0503	4.5642	4.5504	3.8836
	Gradients	2192	564	633	425	3385	3848	5797
50	$\tilde{E}_1(u_k)$ =	4.0600	4.0268	4.0042	3.9572	4.2687	4.1939	3.8675
	Gradients	7419	1146	1217	1264	7974	9216	7553
75	$\tilde{E}_1(u_k)$ =	3.9612	3.9472	3.9384	3.9195	4.1064	4.0401	3.8560
	Gradients	14565	2321	2525	2462	12684	14630	11567
100	$\tilde{E}_1(u_k)$ =	3.9296	3.9231	3.9166	3.9084	4.0093	3.9459	3.8479
	Gradients	20427	4177	5550	5206	17120	19342	16584
150	$\tilde{E}_1(u_k)$ =	3.9055	3.9029	3.9024	3.9014	3.9009	3.8783	3.8416
	Gradients	32651	11725	13379	13587	26617	28600	26385
200	$\tilde{E}_1(u_k)$ =	3.9007	3.9001	3.9001	3.8999	3.8563	3.8524	3.8397
	Gradients	44890	22834	24601	24641	36443	38150	35512
250	$\tilde{E}_1(u_k)$ =	3.8999	3.8997	3.8997	3.8997	3.8444	3.8421	3.8389
	Gradients	56506	33600	35841	35712	46046	47825	45296
300	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8402	3.8392	3.8385
	Gradients	67287	43911	46867	47340	55315	57350	55043
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8386	3.8383	3.8384
	Gradients	78087	54810	58183	57185	65037	67206	65296
400	$\tilde{E}_1(u_k)$ =	3.8996	3.8997	3.8997	3.8996	3.8381	3.8380	3.8384
	Gradients	89041	66297	70480	69090	74787	77312	75117
450	$\tilde{E}_1(u_k)$ =	3.8996	3.8997	3.8997	3.8996	3.8379	3.8379	3.8384
	Gradients	99637	77805	81417	81697	84640	87305	85200
500	$\tilde{E}_1(u_k)$ =	3.8996	3.8997	3.8997	3.8996	3.8378	3.8378	3.8384
	Gradients	104771	89316	94445	93732	94631	97279	95274

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 50\lvert G_1 - 1\rvert$				Norm: $\lVert u \rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	31.88	17.18	7.939	11.77	31.87	13.26	3.910
10	$\tilde{E}_1(u_k)$ =	5.5246	5.0396	4.5328	4.3538	7.7262	4.7182	3.9016
	Gradients	51	79	118	111	101	1194	3679
25	$\tilde{E}_1(u_k)$ =	4.3737	4.4042	4.2144	4.0675	5.7600	4.4010	3.8844
	Gradients	229	279	366	398	982	4176	5438
50	$\tilde{E}_1(u_k)$ =	4.0233	4.0944	3.9969	3.9534	4.7964	4.1257	3.8618
	Gradients	869	823	1067	1277	4965	9483	10578
75	$\tilde{E}_1(u_k)$ =	3.9501	3.9806	3.9408	3.9176	4.3172	4.0238	3.8516
	Gradients	2029	1827	2610	3254	11587	14447	15606
100	$\tilde{E}_1(u_k)$ =	3.9195	3.9314	3.9140	3.9069	4.0762	3.9414	3.8449
	Gradients	4825	3826	5307	7705	17201	19230	20553
150	$\tilde{E}_1(u_k)$ =	3.9014	3.9044	3.9015	3.9004	3.9251	3.8770	3.8406
	Gradients	16045	12904	15806	18054	27341	28813	30294
200	$\tilde{E}_1(u_k)$ =	3.8999	3.9004	3.9000	3.8998	3.8672	3.8513	3.8388
	Gradients	26266	22788	25647	28068	36869	37967	40413
250	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8474	3.8424	3.8383
	Gradients	37982	38763	37001	39422	46193	47799	50581
300	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8411	3.8393	3.8382
	Gradients	49212	50752	48701	50713	55982	57954	60603
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8387	3.8384	3.8381
	Gradients	60012	62038	60573	61558	66482	67887	70408
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8381	3.8380	3.8381
	Gradients	71836	74916	72737	73488	76723	78077	80462
450	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8379	3.8378	3.8381
	Gradients	84958	88719	84563	85359	86896	88300	90705
500	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8378	3.8378	3.8381
	Gradients	99743	101759	96328	100836	96441	98070	100493

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 500\lvert G_1 - 1\rvert$				Norm: $\lVert u \rVert_{L_2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	299.7	138.2	41.62	75.75	299.5	90.01	4.052
10	$\tilde{E}_1(u_k)$ =	5.1060	5.3799	4.4683	5.9917	5.5000	4.2054	4.0099
	Gradients	209	171	244	63	500	2662	3249
25	$\tilde{E}_1(u_k)$ =	4.4486	4.5246	4.1614	4.3789	4.9687	4.0786	3.9655
	Gradients	1679	398	552	243	2548	6080	6540
50	$\tilde{E}_1(u_k)$ =	4.1072	4.0640	3.9767	4.0391	4.4259	3.9690	3.9052
	Gradients	7462	1050	1395	845	7989	11307	11523
75	$\tilde{E}_1(u_k)$ =	3.9821	3.9460	3.9213	3.9368	4.1540	3.9081	3.8765
	Gradients	13309	2311	3231	2436	13545	16158	16713
100	$\tilde{E}_1(u_k)$ =	3.9338	3.9149	3.9068	3.9100	3.9856	3.8731	3.8596
	Gradients	18898	4910	7875	6116	18976	21818	21813
150	$\tilde{E}_1(u_k)$ =	3.9047	3.9012	3.9006	3.9009	3.8828	3.8483	3.8444
	Gradients	309000	16070	17108	17301	29162	32070	31411
200	$\tilde{E}_1(u_k)$ =	3.9002	3.8998	3.8998	3.8998	3.8513	3.8409	3.8397
	Gradients	42643	26396	28555	27360	39309	42272	41282
250	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8420	3.8387	3.8383
	Gradients	54980	39035	39567	39269	48553	52397	51595
300	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8393	3.8381	3.8378
	Gradients	66579	50997	51461	50570	57806	62609	61667
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8383	3.8378	3.8378
	Gradients	79652	62373	63515	62372	67097	73016	72071
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8379	3.8378	3.8378
	Gradients	92926	74259	74711	74614	77286	83769	81800
450	$E_p(u_k)$ =	3.8997	$*$	3.8997	3.8997	3.8378	3.8378	3.8378
	Gradients	10435	$*$	85966	86175	87404	93825	91936
500	$E_p(u_k)$ =	3.8997	$*$	3.8997	3.8997	3.8378	3.8378	3.8378
	Gradients	115604	$*$	96810	96818	97849	104616	102074

6.3.4 Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$

Algorithm 6.51								
$p = 1.0$		E_1				Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$E_p(u_k)$ =	5.255	5.1016	4.537	4.0850	5.253	4.043	3.895
10	$E_p(u_k)$ =	4.3756	4.3817	4.2817	4.0553	4.3925	3.9959	3.8949
	Gradients	10	20	30	52	10	70	29
25	$E_p(u_k)$ =	4.2141	4.2527	4.1574	3.9941	4.1981	3.9415	3.8948
	Gradients	25	36	45	67	25	85	47
50	$E_p(u_k)$ =	4.2009	4.2520	4.1544	3.9914	4.1758	3.9393	3.8948
	Gradients	60	83	99	122	57	141	86
75	$E_p(u_k)$ =	4.2008	4.2510	4.1524	3.9878	4.1753	3.9367	3.8948
	Gradients	88	149	177	209	88	244	144
100	$E_p(u_k)$ =	4.2007	4.2485	4.1492	3.9818	4.1739	3.9329	3.8948
	Gradients	113	215	273	330	113	381	212
150	$E_p(u_k)$ =	4.1997	4.2294	4.1366	3.9619	4.1736	3.9194	3.8948
	Gradients	179	363	555	731	217	824	386
200	$E_p(u_k)$ =	4.1989	4.1691	4.0944	3.9187	4.1727	3.8847	3.8947
	Gradients	309	562	926	1383	351	1480	598
250	$E_p(u_k)$ =	4.1936	3.9691	3.9314	3.9020	4.1679	3.8481	3.8942
	Gradients	449	896	1433	2833	508	2582	836
300	$E_p(u_k)$ =	4.1693	3.9040	3.9015	3.9001	4.1456	3.8406	3.8924
	Gradients	615	1958	3059	6340	683	4744	1152
350	$E_p(u_k)$ =	4.0597	3.8999	3.8998	3.8998	4.0510	3.8388	3.8881
	Gradients	837	7549	10918	15499	916	9753	1603
400	$E_p(u_k)$ =	3.9310	3.8997	3.8997	3.8997	3.9087	3.8382	3.8828
	Gradients	1384	20293	22468	27364	1481	19168	2454
450	$E_p(u_k)$ =	3.9015	3.8997	3.8997	3.8997	3.8550	3.8381	3.8791
	Gradients	2994	32887	32507	39002	2764	27805	3699
500	$E_p(u_k)$ =	3.8999	3.8997	3.8997	3.8997	3.8408	3.8380	3.8759
	Gradients	9626	46310	45139	49871	4843	38048	5179

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 5\lvert G_1 - 1\rvert$				Norm: $\sqrt{\lVert u\rVert_{L_2}^2 + \lVert \nabla u\rVert_{L_2}^2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	5.1031	5.0742	4.572	5.3767	5.102	5.5848	3.895
10	$\tilde{E}_1(u_k)$ =	4.6616	4.8654	4.5072	4.1574	4.7346	4.7094	3.8950
	Gradients	11	23	25	19	10	14	33
25	$\tilde{E}_1(u_k)$ =	4.6217	4.7946	4.4973	4.1094	4.3645	4.1590	3.8944
	Gradients	31	39	57	39	30	35	122
50	$\tilde{E}_1(u_k)$ =	4.5482	4.7902	4.4922	4.0888	4.3300	4.0491	3.8923
	Gradients	57	118	157	100	71	105	304
75	$\tilde{E}_1(u_k)$ =	4.5136	4.7818	4.4876	4.0841	4.2092	4.0415	3.8889
	Gradients	103	226	301	259	121	292	601
100	$\tilde{E}_1(u_k)$ =	4.4312	4.7656	4.4792	4.0783	4.0976	4.0319	3.8857
	Gradients	154	354	470	459	235	615	1068
150	$\tilde{E}_1(u_k)$ =	4.0883	4.6468	4.3972	4.0327	3.9105	3.9135	3.8820
	Gradients	339	645	888	994	834	1454	2298
200	$\tilde{E}_1(u_k)$ =	3.9432	4.0148	4.0689	3.9230	3.8541	3.8417	3.8793
	Gradients	875	958	1307	1697	2365	3254	3967
250	$\tilde{E}_1(u_k)$ =	3.9027	3.9055	3.9025	3.9017	3.8394	3.8382	3.8776
	Gradients	2403	2373	2939	3809	5108	13909	5732
300	$\tilde{E}_1(u_k)$ =	3.9000	3.9000	3.9000	3.9001	3.8382	3.8382	3.8766
	Gradients	9060	8777	11188	10923	17761	26860	7699
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8998	3.8382	3.8382	3.8756
	Gradients	19700	20710	23241	20764	33637	43589	9645
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8998	3.8382	3.8382	3.8747
	Gradients	32639	32477	35824	33269	46062	58188	11605
450	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8998	3.8382	3.8382	3.8737
	Gradients	44348	48257	50286	45801	58337	61206	13549
500	$\tilde{E}_1(u_k)$ =	3.8997	*	*	3.8998	3.8382	3.8382	3.8727
	Gradients	58759	*	*	58412	61430	65043	15816

Algorithm 6.51								
$p = 1.0$		$\tilde{E}_1 = F_1 + 50\lvert G_1 - 1\rvert$				Norm: $\sqrt{\lVert u\rVert_{L_2}^2 + \lVert \nabla u\rVert_{L_2}^2}$		
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	31.88	17.18	7.939	11.77	31.87	13.26	3.910
10	$\tilde{E}_1(u_k)$ =	4.8448	4.9957	4.5370	4.1762	5.0348	4.8971	3.8965
	Gradients	16	18	19	18	15	16	20
25	$\tilde{E}_1(u_k)$ =	4.4333	4.8459	4.5154	4.0949	4.3221	4.0892	3.8942
	Gradients	46	48	50	50	45	46	84
50	$\tilde{E}_1(u_k)$ =	4.2294	4.8358	4.5061	4.0885	4.2396	4.0452	3.8927
	Gradients	109	137	144	155	113	135	265
75	$\tilde{E}_1(u_k)$ =	4.2102	4.8216	4.4972	4.0830	4.2002	4.0425	3.8897
	Gradients	209	241	267	319	213	345	470
100	$\tilde{E}_1(u_k)$ =	4.1678	4.8006	4.4859	4.0761	4.1116	4.0356	3.8854
	Gradients	324	371	426	508	338	578	832
150	$\tilde{E}_1(u_k)$ =	4.0690	4.6496	4.4040	4.0447	3.9131	3.9031	3.8794
	Gradients	609	668	794	890	805	1151	2039
200	$\tilde{E}_1(u_k)$ =	3.9368	4.0116	4.1472	3.9218	3.8468	3.8394	3.9752
	Gradients	1105	999	1187	1373	2584	4136	3639
250	$\tilde{E}_1(u_k)$ =	3.9010	3.9025	3.9041	3.9007	3.8384	3.8382	3.8716
	Gradients	2867	2259	2165	4011	9731	16199	5443
300	$\tilde{E}_1(u_k)$ =	3.8998	3.8999	3.8998	3.8999	3.8380	3.8381	3.8679
	Gradients	12821	11947	12682	13243	24197	30677	7439
350	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8381	3.8643
	Gradients	26902	23841	24785	25256	39296	45368	9441
400	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8381	3.8608
	Gradients	40760	36329	35935	38721	52934	58545	11638
450	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8381	3.8573
	Gradients	53778	47823	50583	51201	67021	71974	13858
500	$\tilde{E}_1(u_k)$ =	3.8997	3.8997	3.8997	3.8997	3.8380	3.8381	3.8541
	Gradients	67745	60355	66512	65067	79750	84124	16291

Algorithm 6.51								
$p = 1.0$ $\quad \tilde{E}_1 = F_1 + 500\lvert G_1 - 1\rvert$ $\quad$ Norm: $\sqrt{\lVert u\rVert_{L_2}^2 + \lVert \nabla u\rVert_{L_2}^2}$								
Step:	dim V_h =	$39 \cdot 39$				$79 \cdot 79$		
k =	u_0 =	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4	u_0^5
0	$\tilde{E}_1(u_k)$ =	299.7	138.2	41.61	75.75	299.5	90.01	4.052
10	$\tilde{E}_1(u_k)$ =	4.8374	5.0026	4.5628	4.3507	4.7766	5.2235	3.9083
	Gradients	16	18	16	18	17	16	20
25	$\tilde{E}_1(u_k)$ =	4.4218	4.9079	4.5376	4.2781	4.4367	4.4181	3.9011
	Gradients	46	48	50	48	47	46	61
50	$\tilde{E}_1(u_k)$ =	4.3492	4.8636	4.5147	4.2018	4.3803	4.1285	3.8931
	Gradients	99	106	125	98	107	137	182
75	$\tilde{E}_1(u_k)$ =	4.3209	4.8418	4.5021	4.0773	4.2741	4.0813	3.8892
	Gradients	179	197	246	192	186	247	398
100	$\tilde{E}_1(u_k)$ =	4.2643	4.8163	4.4868	4.0620	4.1640	4.0348	3.8849
	Gradients	269	319	416	378	302	424	811
150	$\tilde{E}_1(u_k)$ =	4.1042	4.6831	4.4397	3.9187	3.9199	3.8889	3.8806
	Gradients	532	614	947	867	837	966	2074
200	$\tilde{E}_1(u_k)$ =	3.9439	4.4018	4.3845	3.9001	3.8482	3.8408	3.8784
	Gradients	1205	951	1511	7344	2328	2905	3586
250	$\tilde{E}_1(u_k)$ =	3.9015	3.9024	4.1546	3.9000	3.8383	3.8383	3.8765
	Gradients	2689	2313	2106	18941	10311	15376	5207
300	$\tilde{E}_1(u_k)$ =	3.9004	3.8998	3.9047	3.9000	3.8380	3.8382	3.8747
	Gradients	5865	11542	3054	32900	24458	29897	6889
350	$\tilde{E}_1(u_k)$ =	3.9003	3.8997	3.8998	3.8998	3.8380	3.8381	3.8731
	Gradients	7661	26030	11429	26269	37889	43529	8525
400	$\tilde{E}_1(u_k)$ =	3.9003	3.8997	3.8997	3.8997	3.8380	3.8381	3.8713
	Gradients	8194	37586	24186	39302	52344	56323	10307
450	$\tilde{E}_1(u_k)$ =	3.9002	3.8997	3.8997	3.8997	3.8380	3.8381	3.8697
	Gradients	8842	50403	37925	52358	66480	71087	12422
500	$\tilde{E}_1(u_k)$ =	3.9002	3.8997	3.8997	3.8997	3.8380	3.8381	3.8681
	Gradients	9418	63321	61765	67068	78914	83507	14075

$*$: Computer shutdown before the end of the computation.

In the case $dimV_h = 39^2$ we observe always $E_1(u_500) < 3.8999$ and $\tilde{E}_1(u_{500}) < 3.8999$. Typically it even holds $E_1(u_{500}) \leq 3.8997$ and $\tilde{E}_1(u_{500}) \leq 3.8997$. In the case $dimV_h = 79^2$ and $u_0 \in \{u_0^1, u_0^4\}$ it holds mostly $E_1(u_{500}) < 3.8385$ and $\tilde{E}_1(u_{500}) < 3.8382$, except for two cases. In these two cases it appears that the Algorithm 6.51 would find the minimizer if it would compute further. This is indeed the case as the next table will show. Before we come to this table we look at the initial function u_0^5. Often we observe for this initial function that $E_1(u_{500}) > 3.8385$ and $\tilde{E}_1(u_{500}) > 3.8382$. Also here one should let Algorithm 6.51 compute further. But this is very time consuming. For example in the case that we choose the norm $\|\nabla u\|_{L^2}$, the function E_1 and the initial point u_0^5, Algorithm 6.51 computes $E_1(u_{1000}) = 3.8534$ and $E_1(u_{1500}) = 3.8383$, where Algorithm 6.51 needed $20,246$ and $59,246$ gradients respectively. Moreover for this choice the Algorithm 6.51 makes 50 descent steps in a row with

$$E_1(u_{k+1}) - E_1(u_k) < 0.0001$$

before step 67. This makes it hard to formulate stopping criterion for this initial function. For us is was surprising and against our intuition that an approximation of the minimizer of (6.12) on a coarse mesh is not a better initial point than the other two. Mostly u_0^5 is even a worse choice.
Interesting for us was moreover that the dimension of the space has little to no effect on the number of necessary steps to obtain a certain decay. Higher dimension allow even a faster decay in the beginning. Further the number of necessary steps depends only weakly on the choice of function and penalty constant. It only begins to play a role close to the minimizer.
To gain the same values the number of gradients depends only weakly on the the dimension too. Only later, closer to the minimizer, Algorithm 6.51 needs more gradients (per steps), but then Algorithm 6.51 needs by far more gradients per step.
The reason why we also took the functions u_0^2 and u_0^3 into our tables, is that one can see there another interesting effect. First the value is decreased very slowly and then there is a big jump in the value. We mention here that during the minimization process first Algorithm 6.51 essentially rounds off the edges of the characteristic function of the square. Just then Algorithm 6.51 increases substantially the domain of the characteristic functions by rescaling the function in the way that $u_{k+1}(x, y) \approx u_k(\lambda_k x, \lambda_k y)$ for some $\lambda_k \in \mathbb{R}$. Therefore the parameters for Algorithm 6.51 might be "good" in the beginning, but in the long run, they might turn out as a "bad" choice. This makes it so difficult to define and say what "good" or "bad" parameters are.
Next we study roughly a stopping criterion. In smooth optimization one often stops the algorithm in the case that $E_1(u_{k-1}) - E_1(u_k) < \tilde{\varepsilon}$ for some suitable $\tilde{\varepsilon}$ after a descent step. We have seen above that the discretization error is about $3.8997 - 3.7725 = 0.1272$ in the case $dimV_h = 39^2$ and about $3.8378 - 3.7725 = 0.0653$ in the case $dimV_h = 79^2$. Thus one might assume that

$$E_1(u_{k-1}) - E_1(u_k) < 0.0001$$

would be a sufficient stopping criterion. It turns out, that this is not the case. With this stopping criterion Algorithm 6.51 often stops at a point/ function u_k with $E_1(u_k) > 4.0$. Therefore we stop the

algorithm in the case that after 50 descent steps the descent is smaller than 0.0001, i.e. in the case that

$$E_1(u_{k-50}) - E_1(u_k) < 0.0001 \ . \tag{6.57}$$

The following table shows the results for Algorithm 6.51 with this stopping criterion.

Algorithm 6.51							
Norm	dim $V_h =$	$39 \cdot 39$				$79 \cdot 79$	
	$u_0 =$	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4
E_1				Stop if $E_1(u_{k-50}) - E_1(u_k) < 10^{-4}$			
$\|\nabla\cdot\|_{L^2}$	Steps	565	430	398	385	611	462
	$E_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8379	3.8380
	Gradients	25,510	23,258	23,680	22,717	31,097	30,446
	$\|D_k\| =$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-4}$	$5 \cdot 10^{-4}$	$7 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$9 \cdot 10^{-4}$
	$\|D_{k,0}\| =$	0.012	0.13	0.18	0.32	0.0012	0.37
$\|\cdot\|_{L^2}$	Steps	473	491	483	273	572	83
	$E_1(u_k) =$	3.8997	3.8997	3.8998	3.8998	3.8383	4.0426∗
	Gradients	38,179	39,477	39,172	39,310	57,596	16,846
	$\|D_k\| =$	0.0038	0.0018	0.0018	0.12	0.0077	31.0
	$\|D_{k,0}\| =$	0.82	0.45	0.42	39.53	1.58	31.0
$\|\cdot\|_{W^{1,2}}$	Steps	560	423	405	378	639	471
	$E_1(u_k) =$	3.8997	3.8997	3.8997	3.8998	3.8379	3.8380
	Gradients	23,914	25,785	23,870	21,884	31,380	32,516
	$\|D_k\| =$	$2 \cdot 10^{-5}$	$4 \cdot 10^{-4}$	$6 \cdot 10^{-4}$	0.0006	$3 \cdot 10^{-5}$	$8 \cdot 10^{-4}$
	$\|D_{k,0}\| =$	0.012	0.15	0.22	0.39	0.015	.34
$\tilde{E}_1 = F_1 + 5\|G_p - 1\|$				Stop if $\tilde{E}_1(u_{k-50}) - \tilde{E}_1(u_k) < 10^{-4}$			
$\|\nabla\cdot\|_{L^2}$	Steps	369	379	366	371	504	464
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8381	3.8381
	Gradients	24,052	27,149	25,381	26,658	35,420	36,509
	$\|D_k\| =$	0.0012	$7 \cdot 10^{-4}$	0.0014	$1 \cdot 10^{-3}$	0.0018	0.0013
	$\|D_{k,0}\| =$	0.51	0.47	0.56	0.58	0.59	0.64
$\|\cdot\|_{L^2}$	Steps	315	288	282	272	399	388
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8380	3.8379
	Gradients	46,049	42,251	43,031	41,055	57,370	62,220
	$\|D_k\| =$	0.21	0.059	0.19	0.057	0.34	0.37
	$\|D_{k,0}\| =$	49.4	46.3	47.5	53.6	98.4	98.6
$\|\cdot\|_{W^{1,2}}$	Steps	375	375	365	368	522	453
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8998	3.8381	3.8381
	Gradients	26,779	28,901	26,743	25,146	35,394	34,281
	$\|D_k\| =$	0.0011	$6 \cdot 10^{-4}$	0.0016	$9 \cdot 10^{-4}$	0.0014	0.0019
	$\|D_{k,0}\| =$	0.55	0.55	0.51	0.50	0.57	0.43

∗) Note that this in this case $\tilde{E}_1(u_{500}) = 3.8378$ too. Here the stopping criterion is satisfied after only 83 steps.

Algorithm 6.51							
Norm	dim $V_h =$	$39 \cdot 39$				$79 \cdot 79$	
$\tilde{E}_1 = F_1 + 50\lvert G_p - 1\rvert$					Stop if $\tilde{E}_1(u_{k-1}) - \tilde{E}_1(u_k) < 10^{-4}$		
	$u_0 =$	u_0^1	u_0^2	u_0^3	u_0^4	u_0^1	u_0^4
$\tilde{E}_1 = F_1 + 50\lvert G_p - 1\rvert$					Stop if $\tilde{E}_1(u_{k-50}) - \tilde{E}_1(u_k) < 10^{-4}$		
$\lVert\nabla\cdot\rVert_{L^2}$	Steps	354	361	369	354	447	419
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8380	3.8380
	Gradients	25,023	28,604	27,556	26,567	45.263	35,243
	$\lVert D_k\rVert =$	0.0014	$9 \cdot 10^{-4}$	0.0015	$7 \cdot 10^{-4}$	0.0013	0.0016
	$\lVert D_{k,0}\rVert =$	8.68	8.60	8.62	8.60	10.2	8.64
$\lVert\cdot\rVert_{L^2}$	Steps	261	267	274	246	368	364
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8380	3.8379
	Gradients	40,907	42,865	42,833	38,737	61,774	60,822
	$\lVert D_k\rVert =$	0.15	0.19	0.26	0.15	0.54	0.49
	$\lVert D_{k,0}\rVert =$	64,7	71.4	73,7	73.1	120.5	99.3
$\lVert\cdot\rVert_{W^{1,2}}$	Steps	347	355	360	357	463	419
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8380	3.8380
	Gradients	26,154	26,149	27,175	27,626	33,048	35,542
	$\lVert D_k\rVert =$	$1 \cdot 10^{-3}$	0.0013	0.0016	$7 \cdot 10^{-4}$	0.0016	0.0016
	$\lVert D_{k,0}\rVert =$	9.92	8.43	9.89	8.47	8.45	8.43
$\tilde{E}_1 = F_1 + 500\lvert G_p - 1\rvert$					Stop if $\tilde{E}_1(u_{k-50}) - \tilde{E}_1(u_k) < 10^{-4}$		
$\lVert\nabla\cdot\rVert_{L^2}$	Steps	341	352	352	350	459	399
	$\tilde{E}_1(u_k) =$	3.9887	3.8997	3.8997	3.8997	3.8381	3.8381
	Gradients	24,198	25,264	25,811	27,476	30,816	30,728
	$\lVert D_k\rVert =$	0.0027	0.0013	0.0012	0.0018	0.0019	0.0026
	$\lVert D_{k,0}\rVert =$	94.4	92.9	92.9	94.4	94.5	94.5
$\lVert\cdot\rVert_{L^2}$	Steps	229	253	240	257	342	325
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8381	3.8381
	Gradients	39,011	40,003	27,421	41,139	56,529	57,343
	$\lVert D_k\rVert =$	0.16	0.16	0.23	0.91	0.61	0.39
	$\lVert D_{k,0}\rVert =$	499.3	4.92.2	491.5	500.2	502.4	504.0
$\lVert\cdot\rVert_{W^{1,2}}$	Steps	350	345	352	347	451	391
	$\tilde{E}_1(u_k) =$	3.8997	3.8997	3.8997	3.8997	3.8381	3.8381
	Gradients	26,008	24,745	26,566	25,911	32.148	30,817
	$\lVert D_k\rVert =$	0.0021	$9 \cdot 10^{-4}$	0.0011	0.0014	0.0013	0.0015
	$\lVert D_{k,0}\rVert =$	90.7	90.7	90.7	90.7	90.8	90.8

With this stopping criterion Algorithm 6.51 finds almost always a point which appears to be close to the minimizer of (6.12). Only one time Algorithm 6.51 with this stopping criterion stops to early. But as we have seen above, Algorithm 6.51 finds the minimizer if we let it compute longer.
Thus Algorithm 6.51 finds always some function which appears to be the minimizer of (6.12). It seems that the choice of norm and initial function does not effect the number of computed gradients substantially. Never the less we observe that for the L^2-norm often more gradients are computed, but on the other hand less steps are needed. It appears that Algorithm 6.51 needs 1.5 up to 2 times more gradients if the L^2-norm is used instead of one of the others.
In comparison to the above table, where we stop after 500 steps, the dimension of V_h effects stronger the number of steps and of computed gradients till (6.57) is satisfied. This effect is largest for the L^2-norm and smallest for the norm $\|\cdot\|_{W^{1,2}(\Omega)}$. For the norm $\|\cdot\|_{W^{1,2}(\Omega)}$ Algorithm 6.51 needs only about 1.1 times more gradients to satisfy (6.57) if $dimV_h = 79^2$ instead of $dimV_h = 39^2$.

Computations in the case $dimV_h = 159^2$ take very long, because computing one gradient takes very long. Therefore we do not study the convergence in that extend as for the dimensions $dimV_h = 39^2$ and $dimV_h = 79^2$. Also for $dimV_h = 159^2$, Algorithm 6.51 with the L^2-norm computes functions u_{500} with $E_1(u_{500}) < 3.8075$.

Next we have a look at the minimal values of E_1 on V_h computed by Algorithm 6.51, where we vary the dimension. For suitable u_k computed by Algorithm 6.51 we obtain:

u_k computed by Algorithm 6.51			
$n =$	41	81	161
$dimV_h =$	$1,521$	$6,241$	$25,281$
$E_1\Big(u_k\Big) =$	3.8997	3.8378	3.8063
$E_1\Big(u_k\Big) - E_1(\chi_C) =$	0.1272	0.0653	0.0338

It appears that the absolute error $E_1\Big(u_k\Big) - E_1(\chi_C)$ depends linear on the number of grid point on the axes.

6.4 Domains which are Different from the Square

In the following we always consider the energy function

$$E_1 := \frac{F_1}{G_1} \ .$$

As initial function we always choose the solution $u_0 \in V_h$ of the weak equation: For all $v_h \in V_h$ holds

$$\int_{\Omega_h} (\nabla u_0 \cdot \nabla v_h + u_0 v_h - v_h)(x) dx = 0 \ . \tag{6.58}$$

We apply again Algorithm 6.51 and denote by u_k the function produced by Algorithm 6.51 after k steps.

In the case that the Cheeger set C of Ω is known we also study the functions $\chi_C \in BV(\Omega)$ and $\chi_C^a \in V_h$ which is defined by

$$\chi_C^a(x_i) := \chi_C(x_i) \quad \text{for every grid point } x_i \in \overline{\Omega} \ .$$

Thus χ_C is a minimizer of (6.15) for $p = 1$. To get an impression of the discretization error we give the values $E_1\Big(\chi_C\Big)$ and $E_1\Big(\chi_C^a\Big)$ in this case too.

6.4.1 The Triangle

First we consider the triangle given by

$$\Omega := \big\{ t_1(2,0) + t_2(0,2) \,\big|\, t_1,\, t_2 \in]0,1[;\ \text{with}\ \ t_1 + t_2 \leq 1 \big\} \ . \tag{6.59}$$

The Cheeger set of this triangle is easy to compute explicitly, cf. [32, Theorem 3]. With the notation of [32] we compute for our triangle $|\Omega| = 2$, $|\partial\Omega| = 2(2+\sqrt{2})$ and $T(\Omega) = 1 + 2 \cdot \tan\frac{3\pi}{8}$ and we obtain the first eigenvalue

$$\lambda_1 = E_1\Big(\chi_C\Big) = \frac{|\partial\Omega| + \sqrt{|\partial\Omega|^2 - 4(T(\Omega) - \pi)|\Omega|}}{2|\Omega|} \approx 2.9604.$$

The Triangle (6.59)				
Mesh		Algorithm 6.51 with L_2-norm		
Grid points	$E_1\Big(\chi_C^a\Big)$	Steps	Gradients	$E_1(u_{500})$
$38,462$	3.1402	500	$67,339$	2.98813

Figure 10 shows u_{500}.

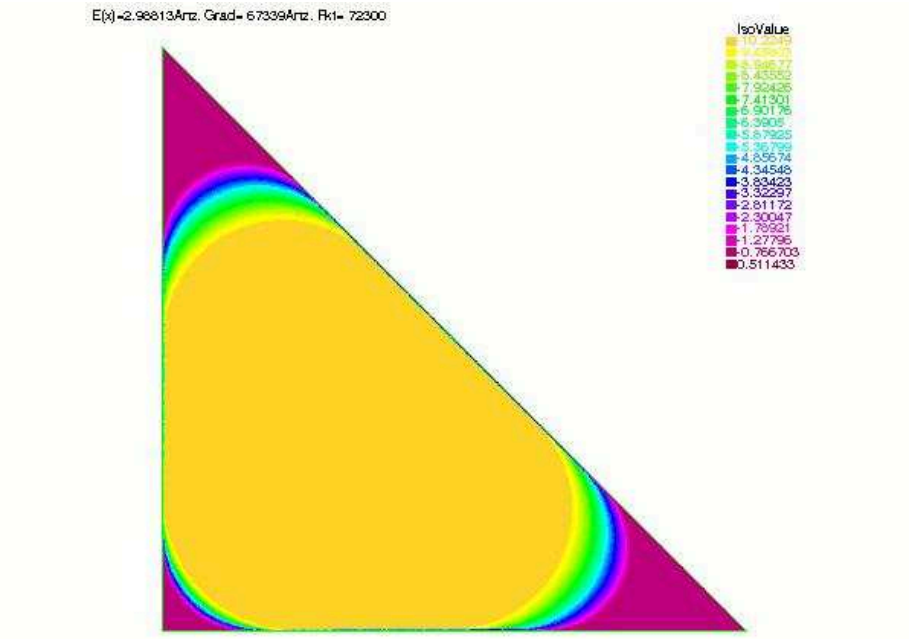

Figure 10: u_{500} of Algorithm 6.51 with L_2-norm for the triangle (6.59).

6.4.2 The Circle

Next we consider the disk Ω of radius 2,

$$\Omega := B_{\mathbb{R}^2}(0\,,\,2)\ , \tag{6.60}$$

which we approximate by a mesh equipped with different numbers of grid points. The Cheeger set C of Ω is the disk Ω it self, so the first eigenvalue is

$$\lambda_1 = E_1\Big(\chi_C\Big) = \frac{2\pi r}{\pi r^2} = 1\ ,$$

cf. [32]. We test Algorithm 6.51 for two meshes and stop after 150 and 300 steps.

The Circle (6.60)				
Mesh		Algorithm 6.51 with L_2-norm		
Grid points	$E_1\Big(\chi_C^a\Big)$	Steps	Gradients	$E_1(u_k)$
14,055	1.0363	150	4,326	1.03267
	1.0363	300	30,898	1.01516
31,491	1.02831	150	4,140	1.07302
	1.02831	300	24,571	1.01308

Figure 11 shows u_{300}. We want to mention that it is possible to create manually and technically a

mesh with $4,264$ grid points such that Algorithm 6.51 with L_2-norm computes some $u_k \in V_h$ with $E_1(u_k) = 1.0002$, cf. Section 6.6.

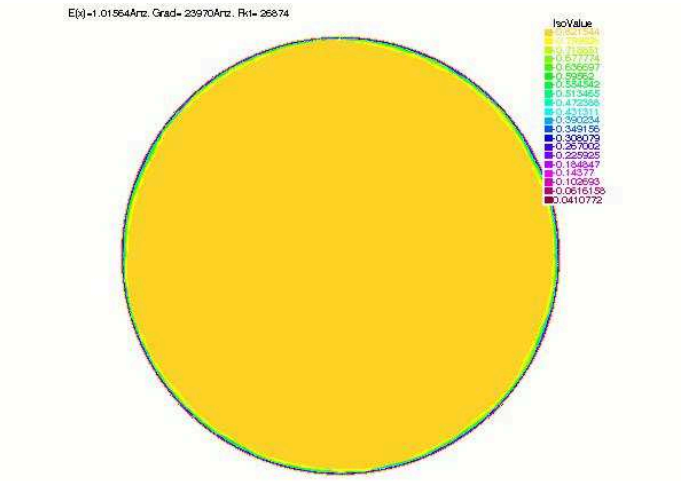

Figure 11: u_{300} of Algorithm 6.51 with L_2-norm for the circle (6.60).

6.4.3 The Oval

Now we consider an oval with elliptic boundary:

$$\Omega := \left\{ (x,y) \in \mathbb{R}^2 \mid \frac{x^2}{5^2} + \frac{y^2}{8^2} < 1 \right\} . \tag{6.61}$$

It is well known that the Cheeger set C of Ω is Ω again, c.f. [32]. There exists no explicit formula for the perimeter of an oval, therefore we compute the first eigenvalue numerically as quotient of the perimeter and the area of Ω:

$$\lambda_1 = E_1\Big(\chi_C\Big) = \frac{P(C)}{A(C)} \approx 0,3294 .$$

We stop Algorithm 6.51 after 350 and 450 steps.

The Oval (6.61)				
Mesh		Algorithm 6.51 with L_2-norm		
Grid points	$E_1\Big(\chi_C^a\Big)$	Steps	Gradients	$E_1(u_k)$
$11,094$	0.3351	350	$4,079$	0.3424
$11,094$	0.3351	450	$16,689$	0.33460

Figure 12 shows u_{350} and u_{450}. We can see that in y direction the approximation u_k of χ_C is not as good as in the x direction. In x direction the ascent u_k to the plateau of χ_C close to the boundary of ∂C is done essentially after just one times the diameter of the triangles. But in the y direction this ascent needs several times the diameter of the triangles.

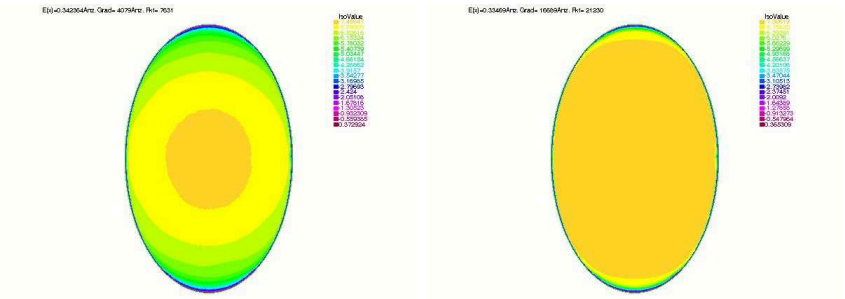

Figure 12: u_{350} and u_{450} of Algorithm 6.51 with L_2-norm for the oval (6.61).

6.4.4 Non Simply Connected Sets

Next we ask the question, what happens if we "cut" an oval out of some oval. Thus we consider a set which is not simply connected. We choose the bounded and open sets Φ_1, Φ_2 and Φ which are given by

$$\begin{aligned}
\Phi_1 &:= \left\{ (x,y) \in \mathbb{R}^2 \mid \frac{x^2}{5^2} + \frac{y^2}{8^2} < 1 \right\} , \\
\Phi_2 &:= \left\{ (x,y) \in \mathbb{R}^2 \mid \frac{x^2}{4^2} + \frac{y^2}{6^2} < 1 \right\} \quad \text{and} \\
\Phi &:= \left\{ (x,y) \in \mathbb{R}^2 \mid \frac{x^2}{3^2} + \frac{y^2}{2^2} < 1 \right\} .
\end{aligned}$$

With this we define the sets

$$\Omega_1 := \Phi_1 \setminus \overline{\Phi} \quad \text{and} \quad \Omega_2 := \Phi_2 \setminus \overline{\Phi} \, . \tag{6.62}$$

For these sets we obtain:

Non Simply Connected Sets (6.62)				
Mesh		Algorithm 6.51 with L_2-norm		
	Grid points	Steps	Gradients	$E_1(u_{600})$
Ω_1	23,849	600	53,208	0.5465
Ω_2	17,602	600	72,526	0.8248

In Figure 13 we see u_{600} for Ω_1 and Ω_2. The crucial point we want to mention here is that u_{600} is one time a function with connected support and the other time it is a function with not connected support. Further it appears to us that Ω_1 has again a unique Cheeger set and the Cheeger set of Ω_1 is Ω_1. Moreover we observe that u_{600} for Ω_2 is quite like a characteristic function in y-direction at the top and at the bottom, in contrast to the situation of the oval (6.61). But, close to the whole, u_{600} does look less like a characteristic function than at the top for Ω_2.

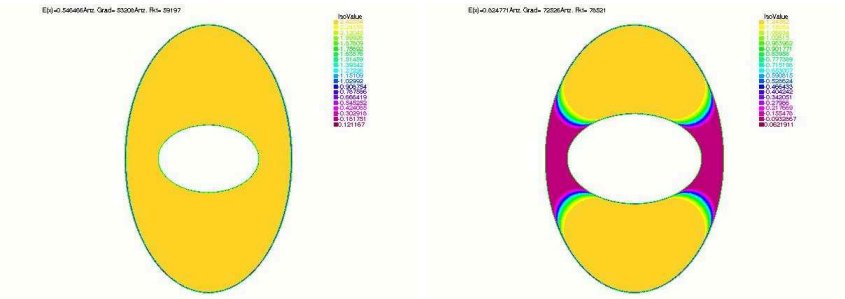

Figure 13: u_{600} of Algorithm 6.51 with L_2-norm for Ω_1 and Ω_2 given in (6.62).

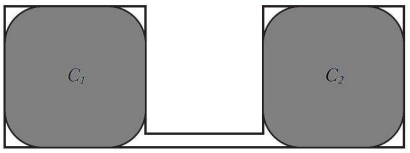

Figure 14: A bar-bell shaped Ω with its Cheeger set $C = C_1 \cup C_2$, cf. [35].

6.4.5 Bar-Bell Shape

Next we consider a simply connected, non convex Ω. In [35] Kawohl and Schuricht gave a bar-bell like Ω such that it admits two disjoint Cheeger sets, cf. Figure 14. We study to different cases. First the bar-bell Ω_1 is symmetric and second the right bar of the bar-bell Ω_2 is stretched. The shapes of the Ω are given e.g. in Figure 15. In the symmetric case of Ω_1 the edges are given by the points

$$(0,0),\ (15,0),\ (15,10),\ (10,10),\ (10,1),\ (5,1),\ (5,10),\ (0,10)$$

and in the nonsymmetric case Ω_2 we just replace the points $(15,0)$ and $(15,10)$ and take the points

$$(0,0),\ (15.5,0),\ (15.5,10),\ (10,10),\ (10,1),\ (5,1),\ (5,10),\ (0,10)\ .$$

Ω_1 has three essentially different Cheeger sets C_1^1, C_1^2 and C_1^3, where C_1^1 and C_1^2 are the Cheeger sets of

$$R_1 := conv\,\{(0,0),\ (5,0),\ (5,10),\ (0,10)\}$$

and

$$R_2 := conv\,\{(10,0),\ (15,0),\ (15,10),\ (10,10)\}$$

respectively and $C_1^3 = C_1^1 \cup C_1^2$. Thus the 1-Laplace operator has on Ω_1 and on R_1 the same first eigenvalue $\lambda_1^{\Omega_1}$. By [32, Theorem 3] with $|R_1| = |R_2| = 50$, $|\partial R_1| = |\partial R_2| = 30$ and $T(R_1) = T(R_2) = 4$ we know that

$$\lambda_1^{\Omega_1} = E_1\Big(\chi_{C_1^1}\Big) = \frac{|\partial R_1| + \sqrt{|\partial R_1|^2 - 4(T(R_1) - \pi)|R_1|}}{2|R_1|} \approx 0.5699.$$

Again by [35] we know that the unique (up to sets of Lebesgue measure 0) Cheeger set C_2^3 of Ω_2 is the Cheeger set of

$$R_3 := conv\left\{(10,0),\ (15.5,0),\ (15.5,10),\ (10,10)\right\}.$$

Thus the 1-Laplace operator has the same first eigenvalue $\lambda_1^{\Omega_2}$ on Ω_2 and on R_3. By [32, Theorem 3] with $|R_3| = 55$, $|\partial R_3| = 31$ and $T(R_3) = 4$ we know that

$$\lambda_1^{\Omega_2} = E_1\left(\chi_{C_2^3}\right) = \frac{|\partial R_3| + \sqrt{|\partial R_3|^2 - 4(T(R_3) - \pi)|R_3|}}{2|R_3|} \approx 0.5344\,.$$

Again we apply Algorithm 6.51 to both sets Ω_1 and Ω_2. We remind that we take as initial function u_0 the solutions of the weak equation (6.58). This has the effect that u_0 is symmetric towards the axis $\{7.5\} \times \mathbb{R}$ for Ω_1. We assume that this is the reason for the almost symmetric approximation u_{400} and u_{800} of the first eigenfunction on Ω_1, cf. Figure 16, but we didn't vary systematically u_0 to study the impact of non symmetric u_0 to the symmetry of the approximation. Of course with our choice of initial function we also get that the initial function for the case Ω_2 is not symmetric. The two initial functions can be seen in the Figure 15.

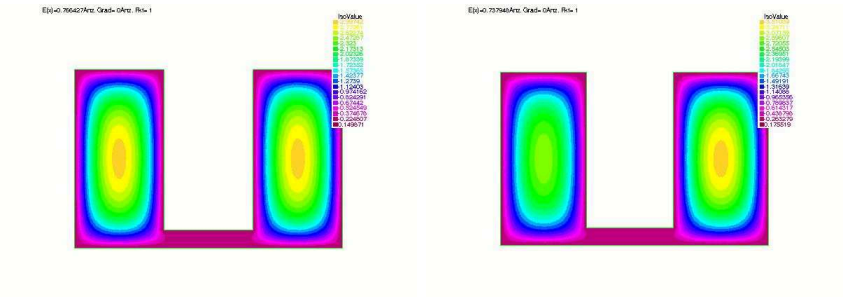

Figure 15: The initial functions u_0 on Ω_1 and Ω_2.

1. First we consider the L_2 norm. We stop after 400 and after 800 steps.

Bar-Bell like Sets					
Mesh			Algorithm 6.51 with L_2-norm		
	Grid points	$E_1\left(\chi_C^a\right)$	Steps	Gradients	$E_1(u_k)$
Ω_1	47,656	0.5934	400	5,012	0.5925
Ω_1	47,656	0.5934	800	26,153	0.5804
Ω_2	49,237	0.5544	400	4,616	0.5777
Ω_2	49,237	0.5544	800	31,197	0.5605

For both domains, we observe that the algorithm produces sequences, which tend first towards a linear combination of the characteristic functions of the Cheeger sets of the rectangles R_1 and R_2 and respectively of the rectangles R_1 and R_3. In Figure 16 we can see u_{400} and u_{800} for Ω_1. We

observe that the approximation seems to stay essentially symmetric, even so the mesh is probably not entirely symmetric. (The mesh is created by a pseudo arbitrary mesh generator.)

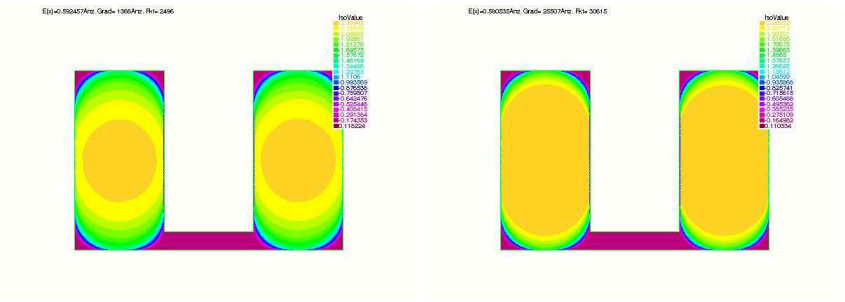

Figure 16: u_{400} and u_{800} of Algorithm 6.51 with L_2-norm for the bar-bell Ω_1.

Next we have a look at Ω_2. First $(u_k)_{k\in\mathbb{N}}$ produced by Algorithm 6.51 approximates a linear combination of $\chi_{C_1^1}$ and $\chi_{C_2^3}$. These approximations become very good. In Figure 17 we see u_{400} and u_{800} for Ω_2. At some point we observe that the algorithm reduces the height of the function u_k on the rectangle R_1 and transfers the entire mass of u_k to the rectangle R_3 as k increases. So finally the u_k produced by Algorithm 6.51 seems to converge $\chi_{C_2^3}$.

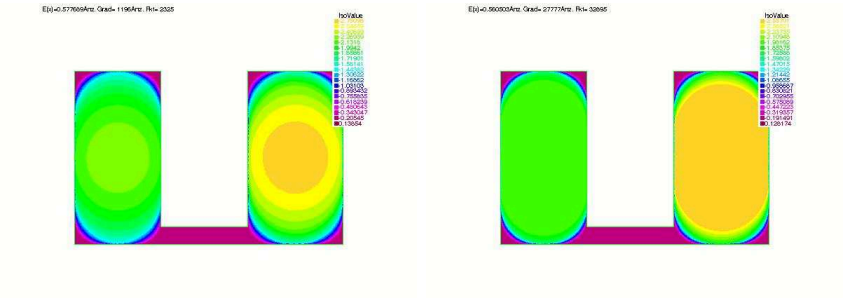

Figure 17: u_{400} and u_{800} of Algorithm 6.51 with L_2-norm for the bar-bell Ω_2.

2. Next we consider again the gradient norm $\|\nabla u\|_{L_2}$.

Bar-Bell like Sets					
Mesh			Algorithm 6.51 with the norm $\|\nabla u\|_{L_2}$		
	Grid points	$E_1\left(\chi_C^a\right)$	Steps	Gradients	$E_1(u_k)$
Ω_1	47,733	0.5985	200	351	0.63007
Ω_1	47,733	0.5985	400	5,154	0.57802
Ω_2	49,090	0.5564	200	391	0.6064
Ω_2	49,090	0.5564	400	4,760	0.543305

(The computations for the two norms had been executed on different computers, therefore the

number of grid points are a little bit different.) We observe the same behavior as for the L_2 norm even much faster. In Figure 18 we see u_{200} and u_{400} on Ω_2.

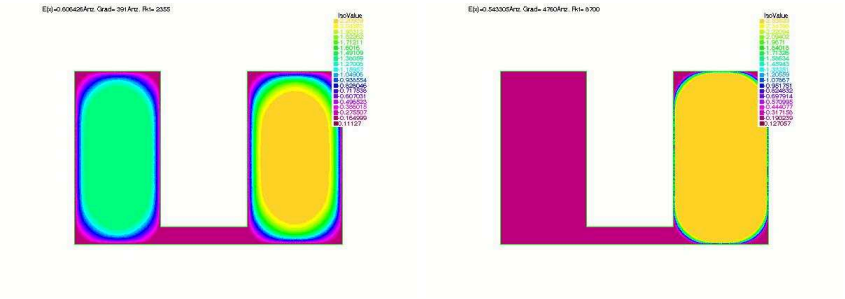

Figure 18: u_{200} and u_{400} of Algorithm 6.51 with the norm $\|\nabla u\|_{L_2}$ for the bar-bell Ω_2.

6.4.6 Not Connected Set

In the situation of the bar-bell we have seen that a connected set might create an eigenfunction, which has not connected support. Now we have a look at a domain, which itself is not connected. To gain an interesting domain, we consider the letters "HSZ", which we write with a very bolt pen. The reason, why we consider these letters is, beside that fact that every letter of these is interesting on its own, is that these are the initials of Hajnal Szakos. In the Hungarian language is "SZ" a letter of its own. The initial function u_0 is shown in Figure 19.

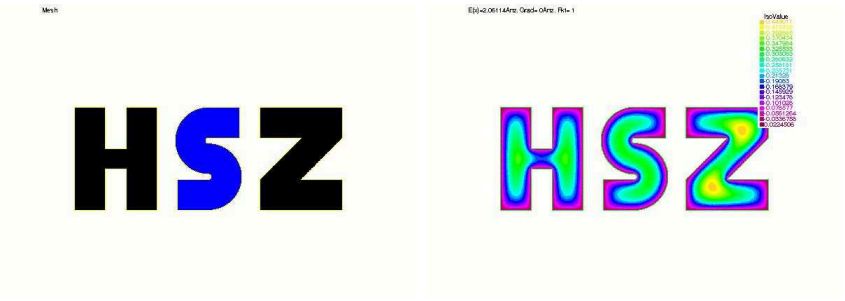

Figure 19: On the left hand side we see the mesh on Ω and on the right hand side we see the initial function on Ω.

HSZ			
Mesh	Algorithm 6.51 with L^2-norm		
Grid points	Steps	Gradients	$E_1(u_{300})$
$15,907$	300	$18,000$	1.525
Mesh	Algorithm 6.51 with the norm $\|\nabla u\|_{L_2}$		
Grid points	Steps	Gradients	$E_1(u_{300})$
$15,907$	300	$4,942$	1.45099

We apply Algorithm 6.51 to the function $E_1 = \frac{F_1}{G_1}$ and consider the norm $\|\nabla u\|_{L_2}$. Again we observe that the sequence computed by Algorithm 6.51 seems to approximate a linear combination of the eigenfunctions of the domains given by the single letters, c.f. Figure 20. After that we again observe that the sequence seems to approximate a first eigenfunction of only one of the letters, namely the letter "Z", compare Figure 22. Observing this effect was quite surprising for us and it is still not clear why this appears. Of course, from the analytical point of view, this is a natural effect, since one can easily prove that a first eigenfunction on "HSZ" must be the continuation by 0 of a first eigenfunction of one of these letters. But numerically it is not clear to us, how the algorithm exchanges the information from one connected component to another. Considering the number of steps, it doesn't look likely that this is an effect caused by not precise computing.

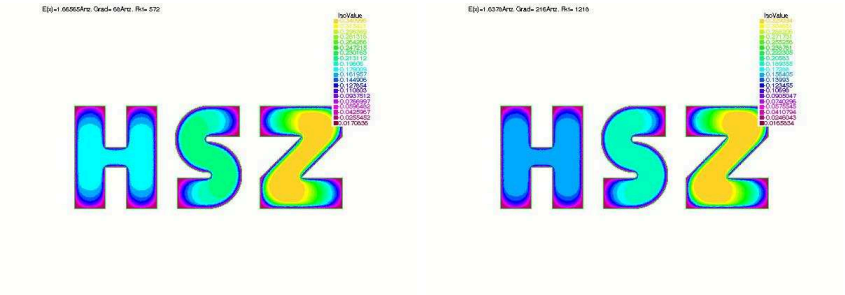

Figure 20: u_{50} (left hand side) and u_{100} (right hand side) of Algorithm 6.51 on HSZ with the norm $\|\nabla u\|_{L_2}$

Next we compare with the L_2 norm. It appears that the solution converges to a linear combination of the Cheeger sets of the single letters, depending on the initial function. For this norm the algorithm does not compute a descent direction, which transfers the mass of the function from the letter "H" and the letter "S" to the letter "Z". In Figure 22 we can see u_{300} of Algorithm 6.51 with the norm $\|\nabla u\|_{L_2}$ and with the L_2. We stopped the computations at this point since they became to time consuming. The results for the norm $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$ look similar to the result for the norm $\|\nabla u\|_{L_2}$.

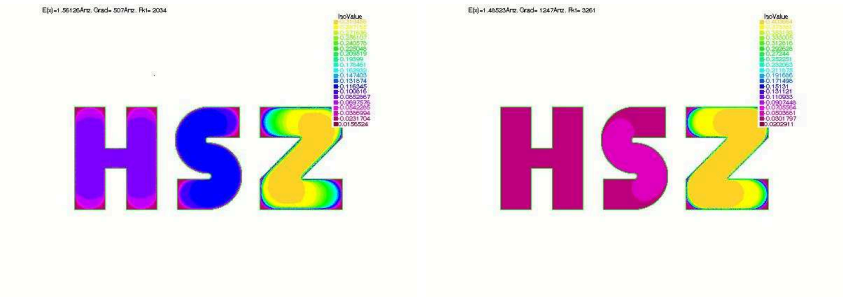

Figure 21: u_{150} (left hand side) and u_{200} (right hand side) of Algorithm 6.51 on HSZ with the norm $\|\nabla u\|_{L_2}$

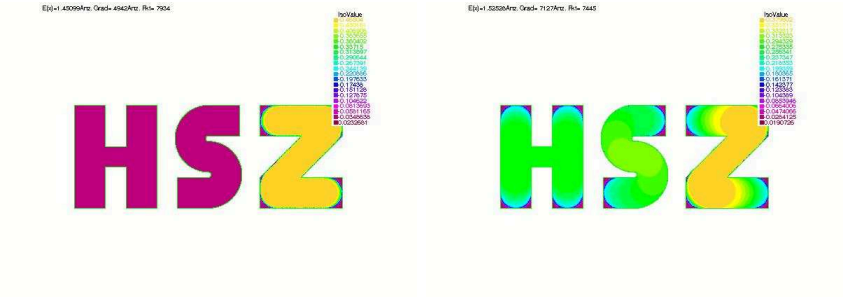

Figure 22: u_{300} of Algorithm 6.51 on HSZ with the norm $\|\nabla u\|_{L_2}$ (left hand side) and with the L^2-norm (right hand side)

6.5 3 Dimensional Domains

Till now we only looked at domains $\Omega \subset \mathbb{R}^2$, since for many domains we know the first eigenfunctions and could compare our results with the analytical solutions. This is not the case for $\Omega \subset \mathbb{R}^3$. Till today, even in the simple case $\Omega =]0,2[^3$, the first eigenfunction is unknown.
For $r > 0$ we define the set

$$\Omega_r^* = \bigcup_{x \in \Omega:\ \mathrm{dist}\,(x, \partial\Omega) > r} B_{\mathbb{R}^n}(x\,,\, r) \tag{6.63}$$

and $\Omega_0^* := \Omega$. Further we define

$$\begin{aligned} I_r &:= \left\{ r \geq 0 \,\middle|\, |\Omega_r^*| \neq 0 \right\} \\ r_m &= \arg\min \left\{ \frac{P(\Omega_r^*)}{A(\Omega_r^*)} \,\middle|\, r \in I_r \right\} , \end{aligned}$$

if r_m exists. In the case $\Omega \subset \mathbb{R}^2$ is convex it holds $r_m = \frac{1}{\lambda_1}$, for the Cheeger set C of Ω holds (up to a set of measure 0)

$$C = \Omega_{r_m}^* \tag{6.64}$$

and χ_C is (up to scaling) the unique minimizer of (6.15), cf.[34]. So the question arises whether the Cheeger set C of a convex set Ω is given by (6.64) in the case $n = 3$ too. Therefore we implemented Algorithm 2.38 also for some $\Omega \subset \mathbb{R}^3$. This way we can further test whether the algorithm also works for $n = 3$. Since the first eigenfunctions are unknown, we can not quantify the quality of our results and so we only present the results, give some estimates and study their appearance. We have already observed in the case $\Omega \subset \mathbb{R}^2$ that we estimate the real minimal value of (6.15) up to about 1 per cent of minimum exact in the case $dim\, V_h = 160^2$.
To get an impression of the discretization error we also look at the functions $\chi^a_{\Omega_r^*} \in V_h$ which are defined by

$$\chi^a_{\Omega_r^*}(x_i) := \chi_{\Omega_r^*}(x_i) \quad \text{for every grid point } x_i \in \overline{\Omega}$$

and give (if known) the values

$$E_1\left(\chi^a_{\Omega^*_{r_m}}\right) \quad \text{and} \quad \min_{r \in I_r} E_1\left(\chi^a_{\Omega_r^*}\right) .$$

Notice, we will not study these values.

We will always minimize

$$E_1 = \frac{F_1}{G_1}$$

and the norm

$$\|\cdot\|_{W_0^{1,1}(\Omega)} \ .$$

The reason for these choices is that we do not know a priori how to choose $K > 0$ of $\tilde{E}_p$. Moreover in the case $n = 2$ this norm gave fast good results for Algorithm 2.38. We apply Algorithm 6.51. To gain an initial function, we simply solve the Poisson equation with constant right side and Dirichlet boundary condition. In the following we always denote by u_k the point/function produced by Algorithm 6.51 after

k steps.

6.5.1 The Cube

First we have a look at the cube $]0,2[^3$, which we construct by defining a mesh on the square $]0,2[^2$ and then construct $]0,2[^3$ with 40 equidistant layers of this mesh. This procedure is a tool of FreeFEM++, we use the command "mesh3 Th3 = buildlayers(Th,MaxLayer,zbound=[zmin,zmax]);".

$\Omega =]0,2[^3$						
Excact	Mesh			Algorithm 6.51		
$E_1\left(\chi_{\Omega^*_{r_m}}\right)$	DoF	$E_1\left(\chi^a_{\Omega^*_{r_m}}\right)$	$\min_{r\in I_r} E_1\left(\chi^a_{\Omega^*_r}\right)$	Steps	Gradients	$E_1(u_{200})$
2.7	$130,831$	3.10	3.046	200	957	2.7910

We follow here two approaches to visualize a function $u_{200} : [0,2]^3 \subset \mathbb{R}^3 \to \mathbb{R}$. First we take a look at the level sets of u_{200} as subsets of $]0,2[^3$, which we indicate by different colors. Second we look at the restriction of the approximating function u_{200} to a layer, i.e. we look at the functions $u_{200}(\cdot,\cdot,0.1*i)$ with $1 \leq i \leq 9$, and visualize their level sets. We can see u_{200} in Figure 23.

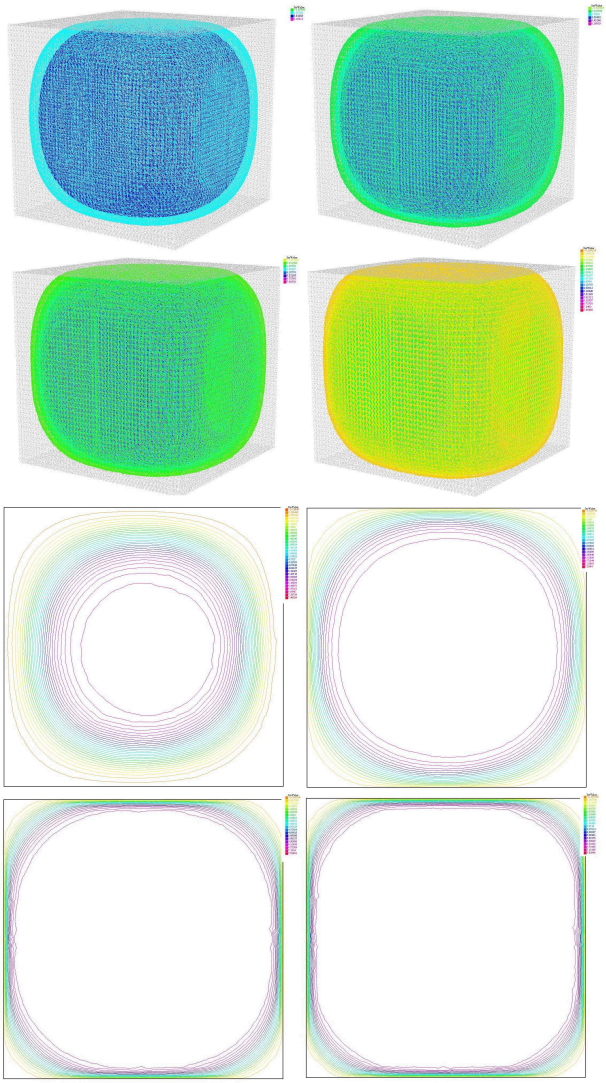

Figure 23: u_{200} computed by Algorithm 6.51 for $\Omega =]0,2[^3$. At the top we see the level sets as subsets of $\mathbb{R}^3$. At the bottom we see the level sets of the restricted functions $u_{200}(\cdot,\cdot,0.1)$, $u_{200}(\cdot,\cdot,0.3)$, $u_{200}(\cdot,\cdot,0.6)$ and $u_{200}(\cdot,\cdot,0.9)$, which map from $\mathbb{R}^2$ into $\mathbb{R}$.

6.5.2 The Parallelepiped

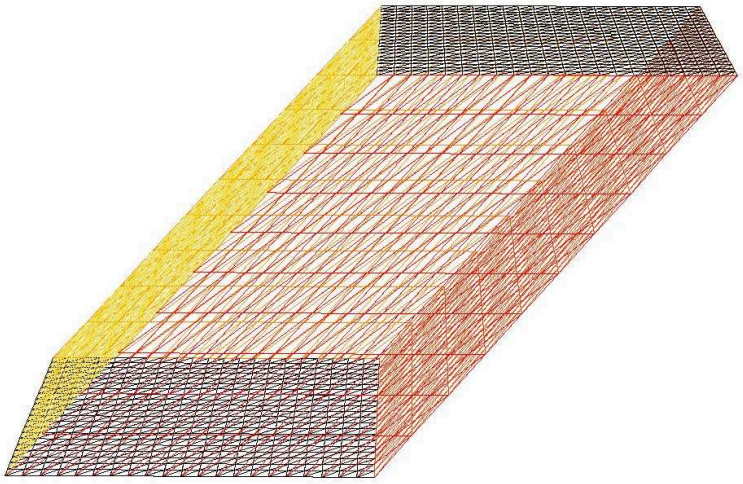

Figure 24: The parallelepiped

The parallelepiped is a convex set too. Therefore it is only natural to assume that the first eigenfunction for this set is a characteristic function of a set, which is the union of all balls with fixed radius and which are included in the parallelepiped. We study the parallelepiped given by the vectors $(2,0,0)$, $(0,2,0)$ and $(0,1,2)$.

$$\Omega = \left\{ s_1(2,0,0) + s_2(0,2,0) + s_3(0,1,2) \,\middle|\, s_1,\, s_2,\, s_3 \in [0,1] \right\} . \tag{6.65}$$

As for the cube, we use the FreeFEM++ command "mesh3 Th3 = buildlayers(Th,...);", with 20 layers, to produce a grid for the parallelepiped. The sets Ω_r^* for the parallelepiped have a more complicated structure than for the cube. Therefore we refrain to determine them.

The Parallelepiped (6.65)			
Mesh	Algorithm 6.51		
DoF	Steps	Gradients	$E_1(u_{200})$
$370,560$	200	$1,026$	3.1363

Figure 25 and Figure 26 show u_{200} computed by Algorithm 6.51. Again our numerical findings support our thesis that the Cheeger set C of a convex set Ω is given by (6.64) in the case $n = 3$ too.

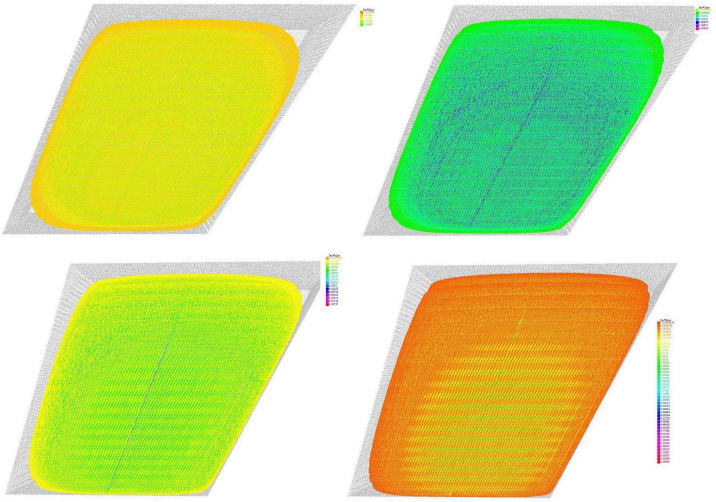

Figure 25: u_{200} of Algorithm 6.51 for the parallelepiped: This figure shows the different level sets of u_{200}.

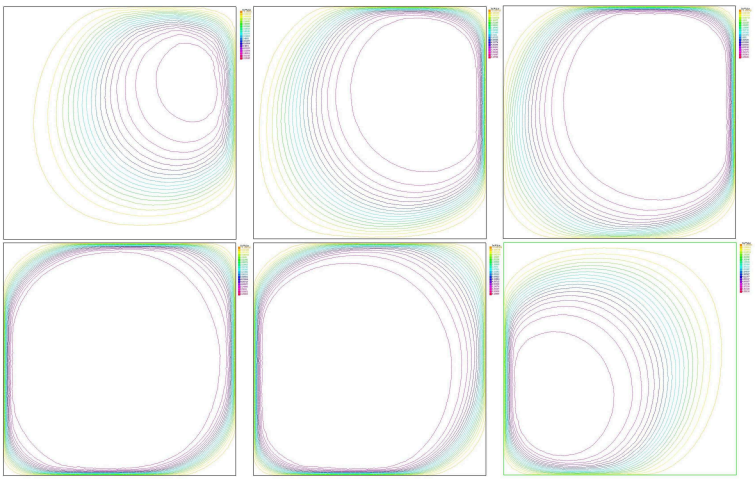

Figure 26: u_{200} of Algorithm 6.51 for the parallelepiped: This figure shows the restricted functions $u_{200}(\cdot,\cdot,0.1)$, $u_{200}(\cdot,\cdot,0.3)$, $u_{200}(\cdot,\cdot,0.5)$, $u_{200}(\cdot,\cdot,0.9)$, $u_{200}(\cdot,\cdot,1.3)$ and $u_{200}(\cdot,\cdot,1.8)$.

6.5.3 Pyramid

We look at the pyramid

$$\Omega = conv\left\{(1,0,0),\ (-1,0,0),\ (0,1,0),\ (0,-1,0),\ (0,0,1)\right\}\ . \tag{6.66}$$

The mesh is created by defining a mesh on the square $[-1,1]^2$ and then using again the command "mesh3 Th3 = buildlayers(Th,...);". Due to the complexity we again refrain to determine Ω_r^*.

The Pyramid (6.66)			
Mesh	Algorithm 6.51		
DoF	Steps	Gradients	$E_1(u_{200})$
51,489	200	986	7.11377

Figure 27 shows u_{200} for the pyramid.

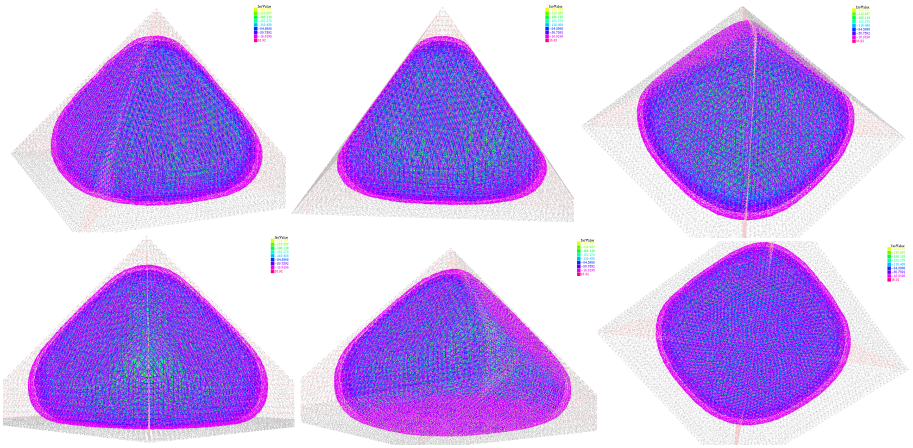

Figure 27: u_{200} of Algorithm 6.51 for (6.66).

6.5.4 The Cylinder

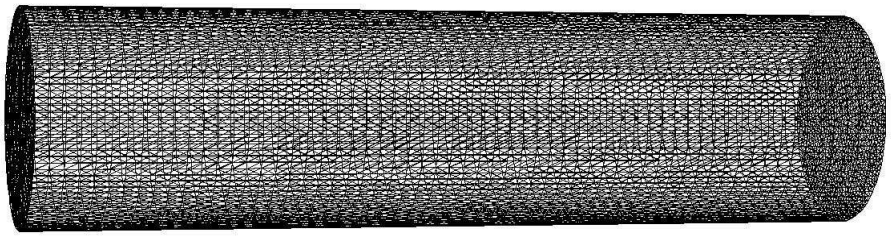

Figure 28: A coarse mesh on the cylinder

Next we study the Cylinder,

$$\Omega_{1,2} := \left\{ \left(x, y, z\right) \in \mathbb{R}^3 \,\middle|\, x^2 + y^2 < 1 \text{ and } z \in]0, 2[\right\} . \tag{6.67}$$

which we gain if we rotate the rectangle

$$R_{1,2} := conv\left\{(0,0,0),\ (1,0,0),\ (0,0,2),\ (1,0,2)\right\}$$

around the z-axis. The mesh is created analogously to the mesh of the pyramid.

As above we compute $r_{\min}$ and $E_1\left(\chi_{\Omega^*_{r_m}}\right)$ numerically, cf. Section 8.1.2.

Exact		Mesh			Algorithm 6.51		
r_m	$E_1\left(\chi_{\Omega^*_{r_m}}\right)$	DOF	$E_1\left(\chi^a_{\Omega^*_{r_m}}\right)$	$\min\limits_{r\in I_r} E_1\left(\chi^a_{\Omega^*_r}\right)$	Steps	Gradients	$E_1(u_k)$
0.50	2.8027	315,000	3.2075	3.1239	200	1336	2.9345

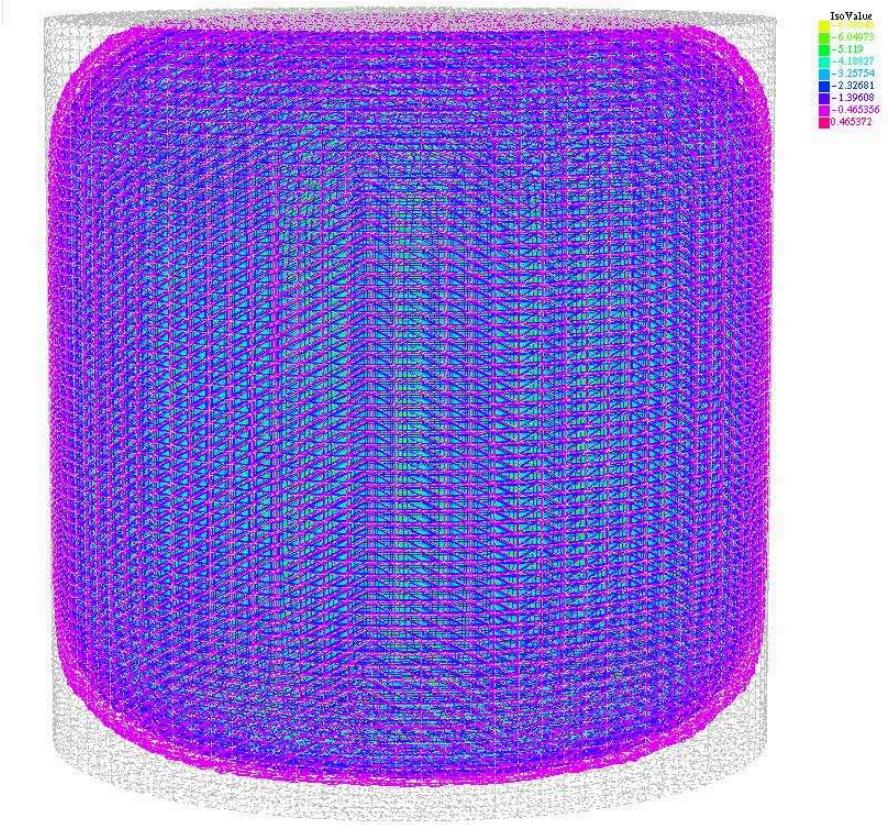

Figure 29: We see u_{200} for (6.67).

6.6 Estimates for the discretization Errors for the 1-Laplace Operator

In Section 6.1.3 we have seen that we can theoretically gain arbitrarily small discretization errors for the minimizer of (6.15). Next we want to make some practical estimates for the discretization error. We study these estimates independently of Algorithm 6.51. We only vary the mesh and the type of approximation of the minimizer of (6.15). In the following these minimizers are (up to scaling) characteristic functions.

Approximations which are Exact on Every Grid Point

Let C be the open Cheeger set of $]0,1[^2$. Next we study the function $\chi_C^a \in V_h$ given by (6.55) and the value $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$ more closely. Notice our quadrature formulae are exact for $G_1(\chi_C^a)$ and $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$, cf. Section 6.1.4. Thus approximation errors are solely due to discretization errors and rounding errors. We will vary the mesh next and study how close $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$ gets to $F_1\Big(\frac{\chi_C}{G_1(\chi_C)}\Big)$ if we use a reasonable amount of grid points. We recall $F_1\Big(\frac{\chi_C}{G_1(\chi_C)}\Big) \approx 3.7725$.

First we create a mesh on $\Omega_\diamond := \overline{C}$. Thus C is the Cheeger set of $\Omega_\diamond$. For fixed $N \in \mathbb{N}$ we create a mesh on $\Omega_\diamond$ by putting equidistantly $N*5$ grid points on every line and $N*10$ grid points on every quarter of a circle, cf. Figure 2. We used the buildmesh command of FreeFEM++. The following table shows the values of $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$ depending on N.

Refinement through increasing the number of grid points on $\partial\Omega_\diamond$							
$N =$	10	20	30	40	50	60	70
vertices	16,680	65,423	147,239	258,016	405,227	579,278	791,847
$F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$	3.8647	3.83544	3.81646	3.81284	3.79648	3.79963	3.79537

We recall that Algorithm 6.51 computed some u_{500} with $E_1(u_{500}) = 3.8071$ on the regular grid with only 25,921 vertices. Thus, the approximation $\frac{\chi_C^a}{G_1(\chi_C^a)}$ is not good, even though the mesh is already highly adapted to the function χ_C.

Of course, one has to admit that we are using way too many grid points in the interior of the Cheeger set C, which are not necessary, since the function is constant there. This motivates the question, what happens if we use refinement techniques to gain a mesh which is fine close to the boundary of C and coarse else. A refinement algorithm has been implemented in "FreeFEM++" and is called "adaptmesh". We define a mesh on the set $\Omega_\square := [-0.1, 1.1]^2$ using the "square" command which creates a mesh similar to the one shown in Figure 8. Starting with an allowed "error" Er_{adapt} of 0.1 we applied several times the "adaptmesh" command to $\Omega_\square$, which adapts the mesh on $\Omega_\square$ along the function χ_C^a allowing just half the "error" of the previous step in each step. Here "error" is a parameter of "adaptmesh". We permitted a maximum number of vertices of 990,000. This way we gained meshes as in Figure 30. On the right hand side of Figure 30 we see χ_C^a for $\Omega_\square$. In the following table we see the values $F_1\Big(\frac{\chi_C^a}{G_1(\chi_C^a)}\Big)$ depending on the degree of refinement.

Refinemend by applying adaptmesh on $\Omega_\square$ to χ_C						
error Er_{adapt}	0.1	0.05	0.025	0.0125	0.00625	0.003125
vertices	166	816	4, 150	28, 393	208, 485	986, 292
$F_1\left(\frac{\chi_C^a}{G_1(\chi_C^a)}\right)$	4.11	4.07	3.93	3.88	3.85	3.85

As we can see the values of $F_1\left(\frac{\chi_C^a}{G_1(\chi_C^a)}\right)$ with $\Omega = \Omega_\square$ are much worse than in the case $\Omega = \Omega_\diamond$,

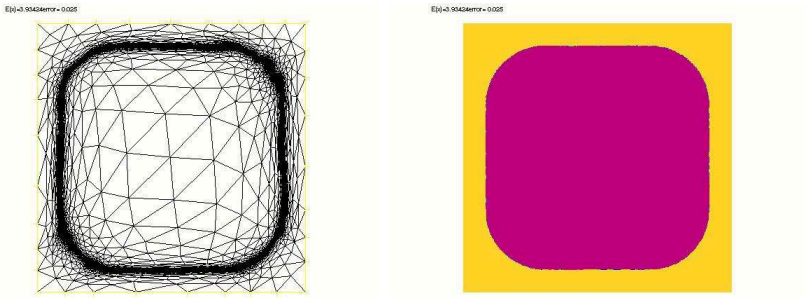

Figure 30: On the left hand side we see the mesh on $\Omega_\square := [-0.1, 1.1]^2$ created by "adaptmesh" with "error" $Er_{adapt} = 0.0125$ using the characteristic function χ_C and on the right hand side we see χ_C^a on $\Omega_\square$.

even though we use similar amounts of grid points.
We conclude that χ_C^a is not the best way to approximate χ_C. Therefore we try another approach.

Analytical Approximation by Continuous Functions

Let U be an open subset of Ω. Next we study approximations of χ_U for a given mesh τ_h, which are of the form

$$\begin{aligned} v_U^\varepsilon(x) &:= \min\{1\,,\,\frac{1}{\varepsilon}\chi_U(x)\cdot dist_{\partial U}(x)\} \qquad (6.68)\\ &= \begin{cases} v_U^\varepsilon(x) = 0 & \text{for} \quad x \notin U\,, \\ v_U^\varepsilon(x) = 1 & \text{for} \quad x \in U \ \text{ with } \ dist_{\partial U}(x) > \varepsilon\,, \\ v_U^\varepsilon = \frac{1}{\varepsilon} dist_{\partial U} & \text{else}\,. \end{cases} \end{aligned}$$

Furthermore we define $v_U^{h,\varepsilon} \in V_h$ through

$$v_U^{h,\varepsilon}(x_i) := v_U^\varepsilon(x_i) \quad \text{for every grid point } x_i \in \overline{\Omega}\,.$$

Note $\varepsilon \neq h$ in general. Next we study $\lim\limits_{\varepsilon\to 0} F_1\left(\frac{v_U^{h,\varepsilon}}{G_1(v_U^{h,\varepsilon})}\right)$ for different meshes.
In Section 6.3.1 we have seen that the approximations of χ_C are much better at the linear parts of ∂C than at the circle like parts of ∂C. For this reason we study two choices of U. First we consider $U_1 := B_{\mathbb{R}^2}(0\,,\,1)$ and second we consider U_2 to be the Cheeger set of $]-0.5, 0.5[^2$. In Section 8.2 we show

analytically that

$$|E_1(v^{\varepsilon}_{U_j}) - E_1(\chi_{U_j})| = o(\varepsilon) \quad \text{and} \quad \left\| v^{\varepsilon}_{U_j} - \chi_{U_j} \right\|_{L_1} = o(\varepsilon) \quad \text{for } j \in \{1, 2\} \ .$$

This makes this approach more natural, than taking χ^a_C, where we do not have similar results for E_1. In the following we study the approximation properties of $v^{h,\varepsilon}_U$ for different meshes. For this purpose we define for fixed $\Omega \subseteq \mathbb{R}^2$ and $U \subseteq \Omega$ the error functions

$$\begin{aligned} Er^{ana} &: \varepsilon \mapsto |E_1(v^{\varepsilon}_U) - E_1(\chi_U)| \ , \\ Er^{num}_1 &: \varepsilon \mapsto |E_1(v^{h,\varepsilon}_U) - E_1(\chi_U)| \text{ and} \\ Er^{num}_2 &: \varepsilon \mapsto |E_1(v^{\varepsilon}_U) - E_1(v^{h,\varepsilon}_U)| \ . \end{aligned}$$

- First we look again at the standard uniform mesh given in Figure 8 which we define on

  $$\Omega_1 := [-1.05, 1.05]^2 \supset U \quad \text{or} \quad \Omega_2 := [-0.55, 0.55]^2 \ .$$

  For U_1 we choose Ω_1 and for U_2 we choose Ω_2. Figure 31 shows the functions Er^{ana}, Er^{num}_1 and Er^{num}_2 for U_1 (left hand side) and U_2 (right hand side), where Ω_1 and Ω_2 have $2561^2 = 6,558,721$ grid points. Thus, e.g. on Ω_2 we choose $h = \frac{1.1}{2560} \approx 0.00043$. These functions look qualitatively the same for different h, so we present only these two graphs here.

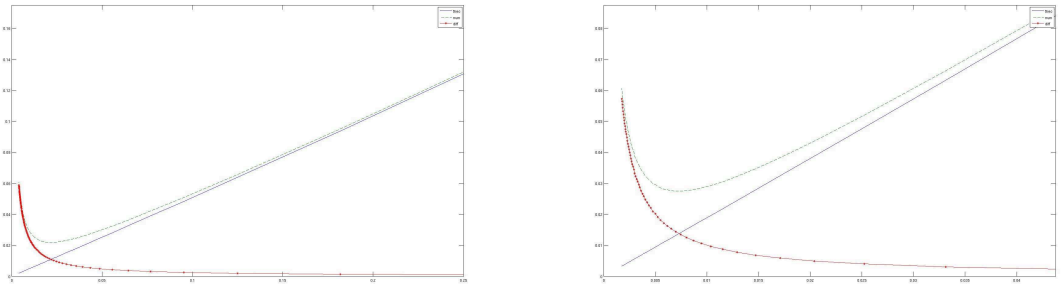

Figure 31: The functions Er^{ana} (blue), Er^{num}_1 (green) and Er^{num}_2 (red) for U_1 (left hand side) and U_2 (right hand side) on a mesh on Ω_1 and Ω_2 respectively with $6,558,721$ grid points.

  They all show that $Er^{num}_2(\varepsilon)$ is almost constant and relatively small for $\varepsilon > \varepsilon^h_0$, where we denote by ε^h_0 the minimizer of Er^{num}_1. But on $]0, \varepsilon^h_0[$ both Er^{num}_1 and Er^{num}_2 are rapidly decreasing functions and Er^{num}_2 is a dominant error term in comparison to Er^{ana}. Therefore we are interested in the functions

  $$h \mapsto \varepsilon^h_0, \quad h \mapsto Er^{ana}(\varepsilon^h_0) \quad \text{and} \quad h \mapsto Er^{num}_1(\varepsilon^h_0) \ ,$$

  which we can see in Figure 32. We can see that all three functions are increasing and they appear to go to zero as h goes to zero. These functions look a bit like stretched square root functions. We observe slow convergence.

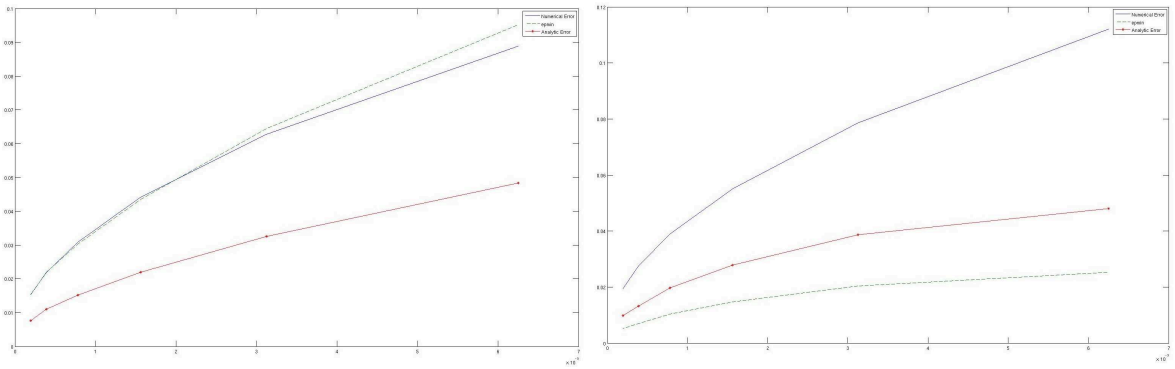

Figure 32: The functions $h \mapsto \varepsilon_0^h$ (broken green line), $h \mapsto Er^{ana}(\varepsilon_0^h)$ (red line with stars) and $h \mapsto Er_1^{num}(\varepsilon_0^h)$ (drawn through blue line) for U_1 (right hand side) and U_2 (left hand side) on a mesh of Ω_1 and Ω_2 respectively with $6,558,721$ grid points.

Moreover we see that for U_1 even for $h = \frac{1}{5120} \approx 1.95 \cdot 10^{-4}$ with $26,224,641$ grid points the absolute numerically error $Er_1^{num}(\varepsilon_0^h) \approx 0.016$. Thus the relative error is approximately 0.008. And the absolute analytical error is still $Er_1^{num}(\varepsilon_0^h) \approx 0.008$. Note that $26,224,641$ grid points is by far too much for our computers.

- Normally we do not know the shape of the (Cheeger) set C, so the above ansatz with a mesh as in Figure 8 is often the best we have. In our computations we even observed more accurate results with a mesh as in Figure 8 than meshes created with the standard refinement algorithm in FreeFEM++ if we use similar numbers of grid points.
  But here we know the shape of C and so we have more possibilities at hand. From our result in Section 6.1.6 we know we can approximate the exact value up to 8 digits with the method mentioned there. So the question arises: Can we create artificially a mesh, which gives better results, if we use the ideas of Section 6.1.6? The answer is yes.
  Now we choose $\Omega_j := U_j$ for $j \in \{1, 2\}$. We observe that U_j is the Cheeger set of Ω_j for $j \in \{1, 2\}$.

  We create the mesh on Ω_j by putting equidistantly $n_1 \in \mathbb{N}$ grid points on ∂U_j and on $(1-\tilde{\varepsilon})\partial U_j$ respectively with $0 < \tilde{\varepsilon} < 0.5$. Further we put equidistantly $n_2 \in \mathbb{N}$ grid points on the circle $(1-2\tilde{\varepsilon})\partial U_j$ with $n_2 << n_1$. On $\overline{U_j} \setminus (1-2\tilde{\varepsilon})U_j$ we allow no further grid points. On $(1-\tilde{\varepsilon})U_j$ FreeFEM++ distributes some additional grid points. In Figure 33 we can see parts of the grid on Ω_1 with $\tilde{\varepsilon} = 0.02$, $n_1 = 201$ and $n_2 = 51$.

	$n_1 =$	$n_2 =$	$\tilde{\varepsilon} =$	Grid points	ε	$Er^{ana}(\tilde{\varepsilon}) \approx$	$Er_1^{num}(\tilde{\varepsilon}) \approx$
U_1	$2,001$	51	0.002	$4,264$	$\varepsilon = \tilde{\varepsilon}$	0.0002	0.0002
U_2	$4,001$	201	0.000075	$11,592$	$\varepsilon = \frac{\tilde{\varepsilon}}{2}$	0.0002	0.0002

To these meshes on Ω_j, which have a form as in Figure 33, we also apply Algorithm 6.51 to E_1 and stop after 500 steps with the function u_{500}. Note U_j is the Cheeger set of Ω_j.

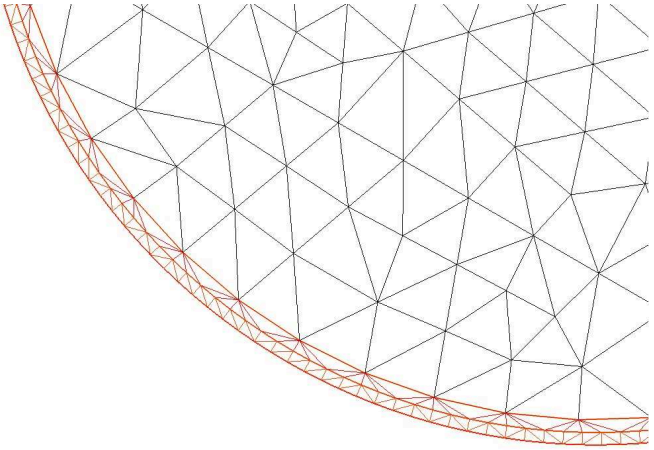

Figure 33: Parts of the grid on Ω_1, with $\varepsilon = 0.02$, $n_1 = 201$ and $n_2 = 51$.

Algorithm 6.51 applied to E_1				
	Gradients	$E_1(u_{500})$	$E_1(u_{500}) - E_1(\chi_U)$	$\frac{E_1(u_{500}) - E_1(\chi_U)}{E_1(\chi_U)}$
U_1	$61,178$	2.00045	0.00045	$2.25 \cdot 10^{-4}$
U_2	$94,859$	3.77284	0.00034	$9 \cdot 10^{-5}$

On grids with a quite small number of grid points we gain very good results. (The meshes have $4,264$ grid points on U_1 and $11,592$ grid points on U_2.) $v_U^{h,\varepsilon}$ approximates χ_U much better than before. On these meshes Algorithm 6.51 gives much better results than in Section 6.3.1 and Section 6.4, although we use less grid points. It is open how this approach can be generalized to the case that the Cheeger set $U_j \neq \Omega_j$. Thus we try a third approach.

- We have seen that the standard refinement with respect to the (limit) function χ_C does not work well. So next we ask what happens if we refine with respect to the analytical approximations v_U^ε. Thus we always start with the same fixed coarse mesh of $\Omega := [-a, a]^2$ as in Figure 8 with 49 grid points. For fixed $\varepsilon > 0$ we use the "adaptmesh" command in FreeFEM++ with the parameter "$nbvx = 990000$" with respect to the function v_U^ε to refine the mesh 6 times where we used the parameter $Er_{adapt} = 0.1 * (0.5)^{k-1}$ in step k. Our results do not depend significantly on the choice of $a \geq 1$. Therefore we take $a := 1$ for U_1 and $a = 0.5$ for U_2. Because with this choice, we can apply Algorithm 6.51 to the resulting mesh for U_2 later. We find the following results:

Refinement by applying adaptmesh to v_U^ε								
$U = U_1$								
$\varepsilon =$	$\frac{0.1}{2}$	$\frac{0.1}{4}$	$\frac{0.1}{8}$	$\frac{0.1}{16}$	$\frac{0.1}{32}$	$\frac{0.1}{64}$	$\frac{0.1}{128}$	$\frac{0.1}{256}$
$dim\, V_h$	$10,964$	$18,370$	$32,229$	$62,812$	$128,364$	$243,323$	$533,365$	$971,020$
$Er^{ana}(\varepsilon)$	0.051	0.025	0.013	0.0063	0.0031	0.0016	0.00078	0.00039
$Er_1^{num}(\varepsilon)$	0.051	0.025	0.013	0.0063	0.0031	0.0015	0.00083	0.00054
$Er_2^{num}(\varepsilon)$	$4 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$5 \cdot 10^{-5}$	0.00015
$U = U_2$								
$\varepsilon =$	$\frac{0.1}{2}$	$\frac{0.1}{4}$	$\frac{0.1}{8}$	$\frac{0.1}{16}$	$\frac{0.1}{32}$	$\frac{0.1}{64}$	$\frac{0.1}{128}$	$\frac{0.1}{256}$
$dim\, V_h$	$5,479$	$8,111$	$12,176$	$20,755$	$38,726$	$81,031$	$155,108$	$335,918$
$Er^{ana}(\varepsilon)$	0.2	0.096	0.048	0.024	0.012	0.0059	0.0029	0.0014
$Er_1^{num}(\varepsilon)$	0.2	0.096	0.048	0.024	0.012	0.0059	0.0029	0.0015
$Er_2^{num}(\varepsilon)$	$6 \cdot 10^{-5}$	$3 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$9 \cdot 10^{-6}$	$1 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$6 \cdot 10^{-5}$

As we can see, $Er_1^{num}(\varepsilon)$ becomes very small compared to the first approach. Again this good approximations where only possible because we know the exact shape of U, which is not the case in general. Finding a method to apply efficiently the above ideas to Algorithm 6.51 in the general case, is left for later work. But we think it is worth to investigate this, as we motivate next.

We apply Algorithm 6.51 to the mesh on $[-0.5, 0.5]^2$ with $\varepsilon = \frac{0.1}{32} = 0.003125$ and $38,726$ grid points. (We could not take a smaller ε, because the computations became to time consuming for our possibilities. This is surprising for us, because for the standard mesh as in Figure 8, we can apply efficiently the algorithm for $321^2 = 103,041$ grid points. The reason is that computing a gradient for the mesh with $38,726$ grid points takes much longer than for the standard mesh.) After 400 steps and about $20,000$ gradients Algorithm 6.51 created an approximation by using the L_2-norm, which has the value 3.779955, thus a relative error of only 0.19. Letting the algorithm compute further we even obtain the value 3.77785.

This value is even smaller than $E_1(v_{U_2}^\varepsilon) = 3.7843$ and by far better than the value 3.8071, which we gained for the standard mesh in Figure 8 with $25,281$ grid points in Section 6.3.1. Thus by changing the mesh to an sophisticated mesh we could decrease the relative error from 0.92 per cent to 0.19 per cent. These findings convince us that we have created a robust and stable algorithm which can solve the problem for proper meshes for the first time, up to our knowledge, and that the main task for the near future is to improve our knowledge about proper meshes for functions $u \in BV(\Omega) \setminus W^{1,1}(\Omega)$ like e.g. characteristic functions. It seems that the approximation errors are dominated by the discretization errors.

Furthermore the fact that even after 400 steps the function value 3.779955 computed by Algorithm 6.51 is already smaller than the value of the analytical approximation $E_1(v_C^\varepsilon) = 3.7843$, makes us believe that Algorithm 6.51 approximates the minimizer on the finite dimensional subspace very well. Algorithm 6.51 gives quite fast approximations, which are even better than those which we implemented naively with our analytical knowledge on the grid. This makes us believe that Algorithm 6.51 finds the minimizer on V_h very well.

6.7 The p-Laplace operator for $p > 1$

As in Section 6.3.1 we test Algorithm 6.51 with $\Omega =]0, a]^2 \quad a \in \{1, 2\}$ for different norms. But this time we choose $p > 1$. We compare Algorithm 6.51 again with our implementations of the steepest descent method (SDM) and the projected gradient method (PGM). Further we compare our results with those of J. Horák in [29]. We recall that all three algorithms are designed for smooth functions. Furthermore, F_p and G_p are Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$, cf. Lemma 8.3.

6.7.1 The Speed of the Algorithm in the Case $p = 1.1$

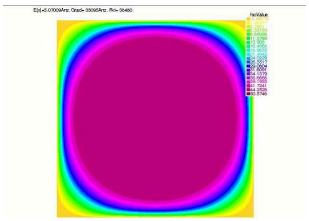

Figure 34: The first eigenfunction of the 1.1-Laplace operator on $]0, 1[^2$.

We consider the case $p = 1.1$ now. We consider exactly the same settings (6.52) except $p = 1.1$ and since the first eigenvalue is larger than 5, we take the functions

$$E_p \quad \text{and} \quad \tilde{E}_p = F_p - K|G_p - 1| \quad \textit{with} \quad K = 10,\ 50 \text{ and } 500\ .$$

In particular we consider the same meshes.
The case $p = 1.1$ was also part of the research of J. Horák , cf. [29]. But there the focus was on gaining the eigenvalue of the p-Laplace operator and not on the rate of convergence. Further it was not mentioned which mesh was used and more importantly, what was the initial function. In contrast to our work the author considered the norm $\|u\|^p := \int\limits_\Omega \left|\nabla u(x)\right|^p (x)dx$ and approximated the predual of the projected gradient of F_p with respect to $G_p(u) = 1$ by an augmented Lagragian method. He had to call the augmented Lagragian method only 10 to 30 times to gain his solutions; but the augmented Lagragian method needed 700 to 2000 iterations to determine one projected gradient of F_p. In every iteration step of the augmented Lagragian method he had to solve equation (6.41), what is essentially computing one element of $\partial E_p(u))$ or $\partial \tilde{E}_p(u))$ for some $u \in V_h$ in Algorithm 6.51. Therefore one might say that the computations in [29] are similar time consuming as computing $7,000$ to $60,000$ elements of $\partial E_p(u)$ or $\partial \tilde{E}_p(u)$ in Algorithm 6.51.
As in Section 6.3.1 we implemented the projected gradient method and the steepest descent method with the three Hilbert space norms (6.47), (6.48) and (6.49) for $p = 1.1$. Since we consider Hilbert spaces, it is an easy task to compute the predual. We first present our results of the projected gradient method, depending on the initial functions and the different norms. This is followed by the steepest descent method for the same norms and initial functions and for the different energy functions and norms. We close the section by showing the results of Algorithm 6.51 for the same norms and functions.

Our implementation of the PGM was stopped as soon as $F_p(u_{k+1}) = F_p(u_k)$ or $k = 50,000$. In the following table, we see the results of PGM. We only consider the case dim $V_h = 39^2$.

Projected Gradient Method (PGM)						
dim $V_h = 39^2$						
Step	Norm: $\|u\|_{L_2}$			Norm: $\|\nabla u\|_{L_2}$		
	$u1$	$u3$	$u4$	$u1$	$u3$	$u4$
0	6.049	6.597	5.9380	6.049	6.597	5.94
5000	5.8870	6.4472	5.8611	5.1064	5.1105	5.1053
10000	5.8801	6.4262	5.8106	5.1039	5.1084	
15000	5.8736	6.4054	5.7673	5.1019	5.1063	
20000	5.8673	6.3851	5.7292	5.1002	5.1043	
25000	5.8612	6.3651	5.6952	5.0987	5.1026	
30000	5.8551	6.3457	5.6645	5.0974	5.1011	
35000	5.8493	6.3267	5.6364	5.0963	5.0997	
40000	5.8436	6.3082	5.6107	5.0953	5.0995	
45000	5.8381	6.2901	5.5869	5.0945		
50000	5.8332	6.2724	5.5648	5.0941		
Step	Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$					
	$u1$	$u3$	$u4$			
0	6.049	6.597	5.94			
5000	5.1056	5.0909	5.1004			
10000	5.1012	5.0909				
15000	5.1010	5.0908				
20000	5.1007	5.0907				
25000	5.1005	5.0907				
30000	5.1002	5.0906				
35000	5.1000	5.0906				
40000	5.0997	5.0905				
45000	5.0995	5.0905				
50000	5.0993	5.0905				

Next the results for our implementation of the steepest descent method. Again we stopped the algorithm as soon as $E_p(u_{k+1}) = E_p(u_k)$ (or $\tilde{E}_p(u_{k+1}) = \tilde{E}_p(u_k)$) or $k = 50,000$.

Steepest Descent Method (SDM)				
$p = 1.1$	dim $V_h = 39^2$			Norm: $\|u\|_{L_2}$
$E_p =$	$\frac{F_p}{G_p}$	$F_p + 10\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	5.648	6.1568	6.917	5.644
Gradients	50000	30000	34000	9
$p = 1.1$	dim $V_h = 39^2$			Norm: $\|\nabla u\|_{L_2}$
$E_p =$	$\frac{F_p}{G_p}$	$F_p + 10\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	5.0906	5.634	5.845	5.887
Gradients	34500	10	8	11
$p = 1.1$	dim $V_h = 39^2$			Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$
$E_p =$	$\frac{F_p}{G_p}$	$F_p + 10\|G_p - 1\|$	$F_p + 50\|G_p - 1\|$	$F_p + 500\|G_p - 1\|$
Value	5.0974	5.632	5.839	5.881
Gradients	50000	6	10	8

Finally we present the results for Algorithm 6.51. As in Section 6.3.1 we stop the Algorithm 6.51 either if $k = 500$ or if the FreeFEM++ routine, which solves (6.41) failed to compute the respresentation for some $b_j \in V_h'$ sufficiently precise in Algorithm 3.9, such that (6.56) didn't hold, even so formally/ theoretically (6.56) should hold.

Algorithm 6.51						
$p = 1.1$		Norm: $\|\nabla u\|_{L_2}$				
		$E_{1.1}$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	6.049	6.597	5.938	6.045	6.773
10	$E_p(u_k)$ =	5.3183	5.3690	5.3804	5.2523	5.2144
	Gradients	10	10	10	10	10
25	$E_p(u_k)$ =	5.2497	5.1861	5.1928	5.1786	5.1744
	Gradients	25	26	25	26	27
50	$E_p(u_k)$ =	5.1836	5.1843	5.1889	5.1772	5.1717
	Gradients	51	55	57	54	55
75	$E_p(u_k)$ =	5.1795	5.1833	5.1883	5.1761	5.1701
	Gradients	81	82	82	81	81
100	$E_p(u_k)$ =	5.1787	5.1824	5.1876	5.1751	5.1689
	Gradients	110	109	109	108	108
150	$E_p(u_k)$ =	5.1777	5.1745	5.1834	5.1734	5.1668
	Gradients	162	163	163	160	162
200	$E_p(u_k)$ =	5.1767	5.1497	5.1626	5.1718	5.1572
	Gradients	215	216	218	214	226
250	$E_p(u_k)$ =	5.1646	5.1274	5.1392	5.1520	5.1233
	Gradients	266	335	333	267	330
300	$E_p(u_k)$ =	5.1316	5.0929	5.0986	5.1204	5.0798
	Gradients	373	551	528	368	521
400	$E_p(u_k)$ =	5.0861	5.0858	*	5.0708	5.0701
	Gradients	1772	11596	*	1328	7312
500	$E_p(u_k)$ =	*	*	*	5.0701	5.0701
	Gradients	*	*	*	19318	13671
		$\tilde{E}_{1.1} = F_{1.1} + 10\|G_{1.1} - 1\|$				
0	$E_p(u_k)$ =	8.493	6.802	8.495	8.490	9.730
10	$E_p(u_k)$ =	5.4200	6.4243	5.8755	5.3911	7.5361
	Gradients	18	17	16	19	19
25	$E_p(u_k)$ =	5.2011	6.3572	5.5901	5.1877	6.6603
	Gradients	48	47	46	49	48
50	$E_p(u_k)$ =	5.1678	6.2084	5.5169	5.1650	6.6346
	Gradients	102	96	95	104	104
75	$E_p(u_k)$ =	5.1610	5.8434	5.475	5.1562	6.5917
	Gradients	170	145	149	174	168
100	$E_p(u_k)$ =	5.1465	5.7049	5.4154	5.1384	6.5117
	Gradients	248	208	219	250	238
150	$E_p(u_k)$ =	5.1043	5.3323	5.1510	5.0879	6.0549
	Gradients	543	357	387	533	392
200	$E_p(u_k)$ =	5.0877	5.0894	5.0857	5.0716	5.1575
	Gradients	4210	941	3479	3948	634
250	$E_p(u_k)$ =	5.0871	5.0867	5.0855	5.0710	5.0706
	Gradients	12123	9898	12535	12987	5350
300	$E_p(u_k)$ =	5.0871	5.0867	5.0855	*	5.0704
	Gradients	17926	15242	16677	*	13590
400	$E_p(u_k)$ =	5.0871	5.0867	5.0855	*	*
	Gradients	18957	16235	17797	*	*
500	$E_p(u_k)$ =	5.0871	5.0867	5.0855	*	*
	Gradients	19257	16535	18106	*	*

Algorithm 6.51						
$p = 1.1$		Norm: $\|\nabla u\|_{L_2}$				
		$\tilde{E}_{1.1} = F_{1.1} + 50\|G_{1.1} - 1\|$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	33.24	9.214	14.91	33.22	16.78
10	$E_p(u_k)$ =	5.6333	6.4384	5.8853	5.6427	7.8481
	Gradients	16	17	16	16	17
25	$E_p(u_k)$ =	5.2557	6.3756	5.6019	5.3001	6.6761
	Gradients	45	47	46	45	47
50	$E_p(u_k)$ =	5.2142	6.2085	5.5750	5.1495	6.5561
	Gradients	95	97	101	100	96
75	$E_p(u_k)$ =	5.1651	5.9448	5.5369	5.1397	6.4861
	Gradients	150	147	164	183	157
100	$E_p(u_k)$ =	5.1471	5.5965	5.4684	5.1274	6.4021
	Gradients	269	205	230	308	228
150	$E_p(u_k)$ =	5.1092	5.2686	5.1908	5.0896	5.9433
	Gradients	2219	368	406	1064	405
200	$E_p(u_k)$ =	5.0975	5.0946	5.0909	5.8185	5.1504
	Gradients	10842	3732	4660	9114	870
250	$E_p(u_k)$ =	5.0975	5.0900	5.0905	5.0819	5.0744
	Gradients	15055	13002	13086	13424	9948
300	$E_p(u_k)$ =	5.0975	5.0900	5.0905	$*$	5.0743
	Gradients	15679	13697	13462	$*$	16321
400	$E_p(u_k)$ =	5.0975	5.0900	5.0905	$*$	5.0743
	Gradients	18023	14492	13729	$*$	16609
500	$E_p(u_k)$ =	5.0975	5.0900	5.0905	$*$	5.0743
	Gradients	19964	14761	14020	$*$	16878
		$\tilde{E}_{1.1} = F_{1.1} + 500\|G_{1.1} - 1\|$				
0	$E_p(u_k)$ =	311.6	36.35	87.10	311.4	96.09
10	$E_p(u_k)$ =	5.7936	6.5023	16	5.6689	7.8585
	Gradients	17	17	5.9244	16	17
25	$E_p(u_k)$ =	5.4750	6.4988	5.6564	5.3911	6.8450
	Gradients	58	65	60	46	47
50	$E_p(u_k)$ =	5.3489	6.3879	5.6564	5.3204	6.7465
	Gradients	131	144	160	96	106
75	$E_p(u_k)$ =	5.2918	6.379	5.6059	5.3081	6.6783
	Gradients	210	194	249	146	235
100	$E_p(u_k)$ =	5.2918	6.3696	5.6059	5.2987	6.6510
	Gradients	310	244	349	196	984
150	$E_p(u_k)$ =	5.2676	6.2605	5.5978	5.1435	6.4949
	Gradients	488	343	542	302	2311
200	$E_p(u_k)$ =	5.2665	6.2046	5.5950	5.1419	6.3390
	Gradients	681	443	737	439	3586
250	$E_p(u_k)$ =	5.2656	6.1875	5.5950	5.1419	6.3369
	Gradients	874	543	937	575	4326
300	$E_p(u_k)$ =	$*$	6.1415	5.5943	5.1418	6.3351
	Gradients	$*$	642	1127	686	5016
400	$E_p(u_k)$ =	$*$	$*$	$*$	5.1417	6.3310
	Gradients	$*$	$*$	$*$	914	7349
500	$E_p(u_k)$ =	$*$	$*$	$*$	$*$	6.3261
	Gradients	$*$	$*$	$*$	$*$	9824

Algorithm 6.51						
$p = 1.1$				Norm: $\|u\|_{L_2}$		
		$E_{1.1}$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	6.049	6.597	5.938	6.045	6.774
10	$E_p(u_k)$ =	5.7617	5.8614	5.3795	5.6181	5.7980
	Gradients	66	37	176	287	853
25	$E_p(u_k)$ =	5.7497	5.8280	5.1680	5.5877	5.3001
	Gradients	83	55	419	324	1483
50	$E_p(u_k)$ =	5.7350	5.7981	5.0990	5.5798	5.1558
	Gradients	113	84	1019	443	2529
75	$E_p(u_k)$ =	7.7181	5.7731	5.0881	5.5708	5.1082
	Gradients	157	120	2576	605	4365
100	$E_p(u_k)$ =	5.6810	5.7447	5.0863	5.5530	5.0824
	Gradients	214	169	6785	809	6775
150	$E_p(u_k)$ =	5.5160	5.5545	5.0862	5.4430	5.0717
	Gradients	423	359	18249	1467	16013
200	$E_p(u_k)$ =	5.1763	5.2117	5.0862	5.2434	5.0703
	Gradients	856	661	29150	2600	25987
250	$E_p(u_k)$ =	5.0911	5.0926	5.0862	5.1307	5.0702
	Gradients	3183	2311	40229	4262	36318
300	$E_p(u_k)$ =	5.0867	5.0867	$*$	5.0917	5.0702
	Gradients	16175	11601	$*$	7678	46386
400	$E_p(u_k)$ =	5.0863	5.0862	$*$	5.0735	5.0702
	Gradients	41532	35070	$*$	21665	64293
500	$E_p(u_k)$ =	5.0863	5.0862	$*$	5.0730	5.0701
	Gradients	61561	56628	$*$	41334	82773
		$\tilde{E}_{1.1} = F_{1.1} + 10\|G_{1.1} - 1\|$				
0	$E_p(u_k)$ =	8.493	6.082	8.495	8.490	9.730
10	$E_p(u_k)$ =	6.3172	5.4857	5.4834	7.5164	6.2059
	Gradients	88	116	95	266	295
25	$E_p(u_k)$ =	5.3345	5.2397	5.2071	5.7082	5.4252
	Gradients	253	327	289	747	953
50	$E_p(u_k)$ =	5.1214	5.1053	5.1029	5.2655	5.2105
	Gradients	989	1084	1042	2376	2537
75	$E_p(u_k)$ =	5.0909	5.0899	5.0901	5.1347	5.1187
	Gradients	2816	2868	2501	5607	6030
100	$E_p(u_k)$ =	5.0869	5.0868	5.0861	5.0946	5.0869
	Gradients	5326	5786	5713	8947	9850
150	$E_p(u_k)$ =	5.0861	5.0861	5.0861	5.0736	5.0738
	Gradients	15585	16130	16458	18317	18487
200	$E_p(u_k)$ =	5.0861	5.0861	5.0861	$*$	5.0708
	Gradients	26490	25589	26147	$*$	29562
250	$E_p(u_k)$ =	$*$	5.0861	$*$	$*$	5.0704
	Gradients	$*$	30557	$*$	$*$	40330
300	$E_p(u_k)$ =	$*$	5.0861	$*$	$*$	5.0704
	Gradients	$*$	31677	$*$	$*$	49691
400	$E_p(u_k)$ =	$*$	5.0861	$*$	$*$	5.0704
	Gradients	$*$	32565	$*$	$*$	69016
500	$E_p(u_k)$ =	$*$	5.0861	$*$	$*$	$*$
	Gradients	$*$	33065	$*$	$*$	$*$

Algorithm 6.51						
$p = 1.1$				Norm: $\|u\|_{L_2}$		
		$\tilde{E}_{1.1} = F_{1.1} + 50\|G_{1.1} - 1\|$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	33.24	9.214	14.91	33.22	16.78
10	$E_p(u_k)$ =	5.6750	5.4454	5.4197	7.6613	8.0453
	Gradients	67	108	116	139	172
25	$E_p(u_k)$ =	5.2382	5.1930	5.1459	5.7210	5.4797
	Gradients	259	344	413	657	730
50	$E_p(u_k)$ =	5.1077	5.0926	5.0906	5.2820	5.2422
	Gradients	994	1434	1678	2446	2863
75	$E_p(u_k)$ =	5.0889	5.0873	5.0867	5.1256	5.1266
	Gradients	3026	4567	5581	6222	6184
100	$E_p(u_k)$ =	5.0865	5.0863	5.0862	5.0863	5.0895
	Gradients	7669	8823	10628	10576	10473
150	$E_p(u_k)$ =	5.0861	5.0862	5.0862	5.0718	5.0719
	Gradients	16948	19362	20225	21462	21965
150	$E_p(u_k)$ =	*	*	*	*	*
	Gradients	*	*	*	*	*
		$\tilde{E}_{1.1} = F_{1.1} + 500\|G_{1.1} - 1\|$				
0	$E_p(u_k)$ =	311.6	36.35	87.10	311.4	96.09
10	$E_p(u_k)$ =	5.6497	5.8004	13.055	8.4533	6.4614
	Gradients	102	91	122	102	351
25	$E_p(u_k)$ =	5.1792	5.2109	5.6604	5.4860	5.3330
	Gradients	416	347	244	686	1129
50	$E_p(u_k)$ =	5.0965	5.0997	5.1225	5.1923	5.1449
	Gradients	2113	1549	1137	2532	3500
75	$E_p(u_k)$ =	5.0886	5.0931	5.0900	5.1089	5.0912
	Gradients	5176	3219	3950	5398	8169
100	$E_p(u_k)$ =	5.0883	5.0923	5.0881	5.0809	5.0758
	Gradients	7217	4975	7875	12372	13452
150	$E_p(u_k)$ =	5.0882	5.0917	5.0877	5.0721	5.0725
	Gradients	10450	7570	11990	24644	24860
200	$E_p(u_k)$ =	5.0881	5.0914	5.0876	5.0717	5.0723
	Gradients	12814	9599	14671	37270	34087
250	$E_p(u_k)$ =	5.0881	5.0914	5.0876	5.0717	5.0723
	Gradients	14983	11248	17025	51501	46630
300	$E_p(u_k)$ =	5.0881	5.0914	5.0876	5.0717	5.0723
	Gradients	16502	12414	18726	66356	58904
350	$E_p(u_k)$ =	5.0881	5.0914	5.0876	5.0717	5.0723
	Gradients	17594	13160	20017	77454	69206
400	$E_p(u_k)$ =	5.0881	5.0914	5.0876	*	5.0723
	Gradients	18361	13814	20905	*	78264
500	$E_p(u_k)$ =	5.0881	5.0914	5.0876	*	*
	Gradients	19617	14903	22097	*	*

Algorithm 6.51						
$p = 1.1$		Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$				
		$E_{1.1}$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	6.049	5.597	5.940	6.045	6.774
10	$E_p(u_k)$ =	5.3197	5.3709	5.2879	5.2524	5.2377
	Gradients	10	10	10	10	10
25	$E_p(u_k)$ =	5.1816	5.1873	5.2184	5.1792	5.1948
	Gradients	25	27	27	26	25
50	$E_p(u_k)$ =	5.1606	5.1855	5.2160	5.1778	5.1454
	Gradients	55	55	54	54	56
75	$E_p(u_k)$ =	5.1604	5.1840	5.2141	5.1766	5.1444
	Gradients	82	82	81	81	83
100	$E_p(u_k)$ =	5.1600	5.1829	5.2123	5.1757	5.1438
	Gradients	109	109	108	108	110
150	$E_p(u_k)$ =	5.1595	5.1796	5.2095	5.1739	5.1425
	Gradients	163	163	162	161	164
200	$E_p(u_k)$ =	5.1541	5.1526	5.1956	5.1724	5.1362
	Gradients	216	221	225	214	239
250	$E_p(u_k)$ =	5.1502	5.1282	5.1538	5.1648	5.1109
	Gradients	290	336	323	277	355
300	$E_p(u_k)$ =	5.1310	5.0925	5.1007	5.1400	5.0773
	Gradients	408	558	507	373	569
400	$E_p(u_k)$ =	5.0862	5.0858	$*$	5.0709	5.0701
	Gradients	1674	15173	$*$	1274	8577
500	$E_p(u_k)$ =	$*$	5.0857	$*$	5.0701	5.0701
	Gradients	$*$	39283	$*$	21547	30883
$\tilde{E}_{1.1} = F_{1.1} + 10\|G_{1.1} - 1\|$						
0	$E_p(u_k)$ =	8.493	6.8023	8.495	8.4902	9.730
10	$E_p(u_k)$ =	5.4093	6.4344	5.6902	5.3848	7.6919
	Gradients	18	16	18	19	18
25	$E_p(u_k)$ =	5.2178	6.3619	5.6439	5.1813	6.6679
	Gradients	46	46	47	49	45
50	$E_p(u_k)$ =	5.1903	6.2047	5.5662	5.1510	5.5994
	Gradients	96	95	96	105	94
75	$E_p(u_k)$ =	5.1653	5.8941	5.4696	5.1444	6.5548
	Gradients	160	144	153	177	159
100	$E_p(u_k)$ =	5.1464	5.7425	5.4143	5.1316	6.4711
	Gradients	235	204	222	256	231
150	$E_p(u_k)$ =	5.1030	5.3445	5.1661	5.0870	6.0081
	Gradients	500	350	387	559	384
200	$E_p(u_k)$ =	5.0874	5.0900	5.0857	5.0716	5.1440
	Gradients	3427	909	3494	4777	627
250	$E_p(u_k)$ =	5.0870	5.0865	5.0854	5.0712	5.0706
	Gradients	11177	9492	12698	13195	5922
300	$E_p(u_k)$ =	5.0870	5.0865	5.0854	$*$	5.0704
	Gradients	16426	14885	18973	$*$	14658
400	$E_p(u_k)$ =	5.8702	5.0865	5.0854	$*$	$*$
	Gradients	17449	15951	20103	$*$	$*$
500	$E_p(u_k)$ =	5.8702	5.0865	5.0854	$*$	$*$
	Gradients	17749	16251	20457	$*$	$*$

Algorithm 6.51						
$p = 1.1$		Norm: $\sqrt{\|u\|_{L_2}^2 + \|\nabla u\|_{L_2}^2}$				
		$\tilde{E}_{1.1} = F_{1.1} + 50\|G_{1.1} - 1\|$				
Step:	dim V_h =	$39 \cdot 39$			$159 \cdot 159$	
k =	u_0 =	u_0^1	u_0^3	u_0^4	u_0^1	u_0^4
0	$E_p(u_k)$ =	33.237	9.2144	14.912	33.219	16.779
10	$E_p(u_k)$ =	5.6318	6.4321	5.7239	5.6349	7.7412
	Gradients	16	18	17	16	18
25	$E_p(u_k)$ =	5.2177	6.3737	5.6330	5.2945	6.9674
	Gradients	45	48	47	46	47
50	$E_p(u_k)$ =	5.1845	6.2063	5.5965	5.1394	6.5049
	Gradients	99	98	102	96	96
75	$E_p(u_k)$ =	5.1752	5.9372	5.5570	5.1325	6.4694
	Gradients	176	148	165	177	149
100	$E_p(u_k)$ =	5.1551	5.6286	5.4797	5.1212	6.3787
	Gradients	291	207	233	298	223
150	$E_p(u_k)$ =	5.1094	5.2530	5.1793	5.0899	5.9886
	Gradients	1838	347	418	2539	401
200	$E_p(u_k)$ =	5.1003	5.0938	5.0911	5.0824	5.1400
	Gradients	8584	4731	5404	9647	983
250	$E_p(u_k)$ =	5.0997	5.0912	5.0905	$*$	5.0892
	Gradients	15399	13922	12551	$*$	8879
300	$E_p(u_k)$ =	5.0997	5.0912	5.0905	$*$	5.0891
	Gradients	19737	15134	14278	$*$	16203
400	$E_p(u_k)$ =	5.0997	5.0912	5.0905	$*$	5.0891
	Gradients	23274	15403	14559	$*$	17300
500	$E_p(u_k)$ =	5.0997	5.0912	5.0905	$*$	5.0891
	Gradients	25656	15657	14821	$*$	17637
		$\tilde{E}_{1.1} = F_{1.1} + 500\|G_{1.1} - 1\|$				
0	$E_p(u_k)$ =	311.6	36.35	87.103	311.3	96.092
10	$E_p(u_k)$ =	5.8181	6.4547	5.7525	5.6428	7.7829
	Gradients	17	18	18	18	18
25	$E_p(u_k)$ =	5.5042	6.4415	5.5688	5.3524	6.8703
	Gradients	46	48	60	47	48
50	$E_p(u_k)$ =	5.3309	6.4365	5.5670	5.2157	6.6024
	Gradients	107	117	150	97	96
75	$E_p(u_k)$ =	5.2709	6.4328	5.5218	5.1584	6.5839
	Gradients	192	180	239	147	799
100	$E_p(u_k)$ =	5.2708	6.3681	5.5218	5.0973	6.5784
	Gradients	292	239	339	197	2223
150	$E_p(u_k)$ =	5.2525	6.2807	5.5052	5.0941	6.5679
	Gradients	481	339	530	340	3471
200	$E_p(u_k)$ =	5.2493	6.2614	5.5018	5.0941	6.5575
	Gradients	672	439	723	490	4310
250	$E_p(u_k)$ =	5.2484	6.2434	5.5011	5.0941	6.5416
	Gradients	865	539	918	640	5656
300	$E_p(u_k)$ =	$*$	6.1406	$*$	5.0941	6.5326
	Gradients	$*$	638	$*$	774	7961
400	$E_p(u_k)$ =	$*$	6.0750	$*$	5.0941	6.4169
	Gradients	$*$	838	$*$	974	10103
500	$E_p(u_k)$ =	$*$	5.9631	$*$	5.0940	6.4129
	Gradients	$*$	1037	$*$	1194	12495

As we can see again Algorithm 6.51 appears to be a better choice to compute the minimal function of E_p and $\tilde{E}_p$ than SDM and PGM. Not only does it get by far closer to the minimal value than the PGM and the SDM on the given meshes, it has to solve the equation (6.41) substantially less times for the same norms. Compared to these methods the algorithm appears to be quite reliable in the sense that for the functions $\frac{F_p}{G_p}$ and $F_p + 10|G_p - 1|$ the Algorithm 6.51 always reaches values less than 5.0871 in the case $\dim V_h = 39^2$ and values less than 5.072 in the case $\dim V_h = 159^2$. For the two other functions $F_p + 50|G_p - 1|$ and $F_p + 500|G_p - 1|$ the results look a bit different. Except for the L_2 norm we end up with larger values and it appears that the iterations points of Algorithm 6.51 does not converge to the minimal function on the mesh. In general it seems that we always only reach functions close to the minimal function. We assume that the main reason for that behavior is the fact that we do not compute the value $G_p(u)$ for $u \in V_h$ sufficiently exact, cf. Section 6.1.4. The indicator which leads us to this assumption is the observation that Algorithm 6.51 is so often aborted. As in the case $p = 1$ the algorithm is aborted in the case that the Algorithm 3.9 failed to find the proclaimed gradient on the line segment, because the element $b_j \in \partial E_p(\xi)$ (or $b_j \in \partial \tilde{E}_p(\xi)$) wasn't computed sufficiently exact for some $\xi \in V_h$. But here we are even in the smooth case, which means Algorithm 3.9 stops with a proper gradient by Proposition 3.17 if we would compute exactly. So we can conclude that we do not compute the gradients precise enough on these line segments and so we assume that we do not compute the gradients in other functions precise enough too. But we can also not exclude that the algorithm finds a critical point, which is not a minimizer.
Finally we compare shortly our results to the results of [29]. As mentioned (after rescaling)[27] the author needed to solve $7,000$ up to $60,000$ times equation (6.41) to gain the value $2.3649 \cdot 2^{1.1} \approx 5.0693$. The mesh he used had $83,968$ grid points. If we use our standard grid with $290^2 = 84,100$ grid points, the norm $\|\nabla u\|_{L_2}$, the energy function $E_p = \frac{F_p}{G_p}$ and the initial function u_0^1 we need $6,839$ gradients to gain the value 5.06936 and if we use the initial function u_4 we need $8,816$ gradients to reach the value 5.06933. So in the good case we are as good as in the approach in [29].

6.7.2 The Speed of the Algorithm in the Case $p = 10$

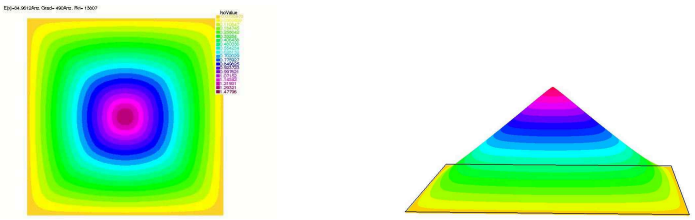

Figure 35: The first eigenfunction of the 10-Laplace operator on $]0, 2[^2$.

Next we choose $p = 10$. Here we consider the square $\Omega := [0, 2]^2$ instead of $[0, 1]^2$ to avoid rounding errors, which have a huge impact for the power 10. Again we point out that we do not compute G_{10}

[27] In [29] the author considered the square $[0, 2]^2$, so we have to multiply his approximated first eigenvalue by $2^{1.1}$

exact. If we are talking about $G_{10}(u)$ for $u \in V_h$ we mean in fact $G_{10}^a(u)$. This approximation makes the Algorithm 3.9 a bit unreliable. Nevertheless we obtain astonishing good results.
We mention again [29], where J. Horák studied the case $p = 10$ with the augmented gradient method and projected gradients. In this paper he needed 10 to 30 projected gradients of F_{10} to obtain the value 34.9900 on a mesh with $83,968$ points. We do not know how the mesh was precisely created, but it is for sure not our standard mesh, since $\sqrt{83,968} \approx 289.77 \notin \mathbb{N}$. Further we point out that for every determination of a gradient, the augmented gradient method needed $1,200$ to $3,000$ iterations. In every iteration of the augmented gradient method he had to solve the Poisson equation. Solving the Poisson equation is the time consuming part in computing an element of $\partial E_{10}(u)$ (or $\partial \tilde{E}_{10}$). Thus we might say one iteration of the augmented gradient method is similar time consuming as one computation an element of $\partial E_{10}(u)$ (or $\partial \tilde{E}_{10}$) in Algorithm 6.51. So with this calculation the algorithm in [29] needed a similar amount of computational time as $12,000$ to $90,000$ computations of an element of $\partial E_p(u)$ or $\partial \tilde{E}_p(u)$ for some $u \in V_h$ in Algorithm 6.51, if we understand the author right. We do not know the initial point used in [29].
We choose our standard mesh, cf. Figure 8, with $291^2 = 84,681$ grid points and the initial function $u_0 \in V_h$ defined by

$$u_0(x_i, y_i) := \sin\left(x_i \frac{\pi}{2}\right) \sin\left(y_i \frac{\pi}{2}\right) \quad \text{for every grid point } (x_i, y_i \in \overline{\Omega}\,.$$

We compare Algorithm 6.51 with our implementations of the simple steepest descent method and the projected gradient method. As above we take both functions E_{10} and $\tilde{E}_{10}$ and all three Hilbert space norms on V_h given above. Since we know from [29] that the minimal eigenvalue is less or equal 34.99 we choose $K = 50$. We stop the algorithms, if $E_{10}(u_k) - E_{10}(u_{k-10}) < 0.0001$ (or respectively if $\tilde{E}_{10}(u_k) - \tilde{E}_{10}(u_{k-10}) < 0.0001$ or $F_{10}(u_k) - F_{10}(u_{k-10}) < 0.0001$).
The algorithms give the following results:

Norm:	$\|\nabla\cdot\|_2$		$\|\cdot\|_2$		$\sqrt{\|\cdot\|_2^2+\|\nabla\cdot\|_2^2}$	
Function:	E_{10}	$\tilde{E}_{10}$	E_{10}	$\tilde{E}_{10}$	E_{10}	$\tilde{E}_{10}$
Steepest Descent Algorithm						
Step k:	500	47	500	500	500	39
Value at u_k:	34.9943	35.90	94.6403	68.03	34.9943	35.72
Step k:	1,000	*	1,000	1,000	1,000	*
Value at u_k:	34.9938	*	87.3145	67.21	34.9938	*
Step k:	2,250	*	2,975	2,000	2,300	*
Value at u_k:	34.9936	*	68.2975	65.96	34.9936	*
Projected Gradient Method						
Step k:	500		500		500	
Value at u_k:	34.9943		128.519		34.9943	
Step k:	1,000		1,000		800	
Value at u_k:	34.9938		125.553		34.9939	
Algorithm 6.51						
Step$k$:	300	300	115	100	300	300
Value at u_k:	34.9948	34.9943	76.91	64.00	34.9949	35.01
Gradients :	319	762	2,520	11,431	319	853
Step k:	410	500	230	*	410	500
Value at u_k:	34.9937	34.9939	55.70	*	34.9936	34.9938
Gradients :	704	1,178	5,619	*	716	1,453

(*: The algorithm was stopped without result.)
We observe that all algorithms are similar fast. But for the L^2-norm all algorithms seem to fail. We assume that the gradients of E_p and $\tilde{E}_p$ are not computed properly.
As pointed out above we have to be careful with this results, since the minimal value depends strongly on the mesh and since we approximate G_p quite rough. To point out the depending of the results on the mesh, we consider the following algorithm.

1. Create the mesh on $[0,2]^2$ as in Figure 8 with 41^2 grid points.
2. Make 60 steps with Algorithm 6.51.
3. Adapt the mesh along the result of Algorithm 6.51 with the command "Omega=adaptmesh(Omega,xk,err=.0001,nbvx=15000);"
4. Make again 60 steps with Algorithm 6.51.
5. Adapt the mesh along the result of Algorithm 6.51 with the command "Omega=adaptmesh(Omega,xk,err=.00005,nbvx=90000);"
6. Make again 60 steps with Algorithm 6.51.

After the first time applying Algorithm 6.51 we obtain after 115 gradient calculations the value $E_{10}(xk) = 36.532$ on a mesh with 1,764 vertices. After the second time applying Algorithm 6.51 we obtain after

$362 = 115 + 247$ gradient calculations the value $E_{10}(xk) = 35.0232$ on a mesh with $15,000$ vertices. And after the third time applying Algorithm 6.51 we obtain after $538 = 362 + 176$ gradient calculations the value $E_{10}(xk) = 34.9617$ on a mesh with $75,044$ vertices. This tells us that also in the paper [29] the mesh was not chosen optimal, since we can obtain with less grid point a better result. Further it appears that for $p = 10$ the standard mesh refinement techniques work well. Again we observe that Algorithm 6.51 finds the minimizer of (6.36) in the case that SDM or PGM find this minimizer.

7 Conclusion and Outlook

We have developed a calculus for gradients of locally Lipschitz continuous functions on sets. We have used these gradients to define optimal descent directions on sets. We proved that many results from smooth analyis can be generalized naturally to this calculus. This theory is a useful achievement on its own.
Then we applied this theory to gain a robust and fast descent algorithm for nonsmooth nonconvex functions. This algorithm was presented as composition of an outer and an inner algorithm. We proved that every accumulation point of every sequence produced by the entire algorithm (or the outer algorithm) is a critical point in the sense of Clarke. Under mild additional assumption, we could even prove that these sequences are convergent to some critical point in the sense of Clarke. The outer algorithm was formulated for locally Lipschitz continuous functions $f : X \to \mathbb{R}$, where X is a reflexive, strictly convex Banach space such that X' is strictly convex too. The inner algorithm requires that X satisfies Clarkson's inequalities. We formulated the inner algorithm for subspace of $W^{1,p}(\Omega)$ with $1 < p < \infty$. Thus it is left to the user whether he follows the first-discretize-then-optimize or the first-optimize-then-discretize approach.
After that we showed that our (entire) algorithm can be seen as globalized nonsmooth Newton method under reasonable assumptions. In particular, we could prove that our (entire) algorithm converges superlinearly under mild assumptions on the gradient of the energy function. This is a property, which we hope to exploit for elastic contact problems with friction in later work. We have tested it for models like Mooney-Rivilin and Neo-Hook's law. These results were very promising. We give them in later work.
Furthermore, we have tested our algorithm with several benchmark problems. There we could see that our algorithm is fast and robust. It can solve every minimization problem, which the other algorithms under consideration solve. Most of the time, it was more robust, faster and it could handle more functions on higher dimensional Banach spaces. Our globalized nonsmooth Newton method showed very promising results too.
At last we tested our algorithm with the p-Laplace operator for $p \geq 1$ with the focus on the case $p = 1$. For the first time, we could compute the first eigenfunction of the 1-Laplace operator numerically with FEM. It appeared that we could compute these eigenfunctions robustly for many different domains up to the discretization error. The algorithm seems to work well even in the case $\Omega \subseteq \mathbb{R}^3$. Note that the first eigenfunctions of the $1-$Laplace operator are unkown in this case even for the cube. We additionally analyised the speed of our algorithm on the square extensively. Our algorithm found for all initial functions, all different Hilbert space norms and the different energy functions E_1 and $\tilde{E}_1$ robust and fast the minimizer in the case $p = 1$. In the case $p \geq 1.1$ we compared our algorithm with an existing one by J. Horák . Here the results depend strongly on the norm.

Outlook

At last we give an outlook.

- The most important problem we have to solve in the future, is to find a suitable abort criterion.

As we can see in the computations the amount of the descent is not decreasing during the computations. We often observe in Section 6 that the decay is low for several steps and suddenly within 100 steps the approximation is close to something that appears to be a minimal point. (E.g in the case that we apply Algorithm 6.51 to $\tilde{E}_1$ with the norm $\|\nabla u\|$, initial function u_0^1 and $K = 5$ we even observe no descent larger than 10^{-4} between step 25 and 50. Then the decay becomes larger.) And after that the decay in every descent step is very small again. Then Algorithm 6.51 investes a lot computational time for very low decay, which is often less than 10^{-4} over 100 steps. The question arises when should we stop the algorithm. Probably we should do it earlier and clearly our abort criterion, to stop after 500 steps, is not the best. The stopping criterion to stop in the case of $E_1(u_{k-49}) - E_1(u_{k+1}) < 0.0001$ is by far better than stopping in step 500. But it is also not obtimal as the computation for E_1 with the L^2-norm and initial function u_0^4 shows.
A criterion, which works quite well for the benchmark problems above, is to stop iff ε_k is sufficient small and $\|D_k\|$ is sufficient small. But in contrast to those benchmark problems, where we considered the Euclidean norm, we have difficulties to define "sufficient small" in the case of the p-Laplace operator for the three different norms, which we are using here. We would need some a priory estimates about the norms of which we do not have knowledge yet.

- Further we have seen that the discretization error is realively large for eigenfunctions of the p-Laplace operator for $p \geq 1$. We could decrease this error by choosing the mesh suitable, but we had to do it manually. Thus a further task for the futur will be to find better mesh generators or mesh refinement algorithms for the 1-Laplace operator.

- Moreover we would like to apply systematically our algorithm to elastic contact problems with friction. As mention we gained promesing results for the Mooney-Rivlin model and Neo-Hook's law with Tresca or Coulomb friction. We would like to compare our algorithm with other algorithms for this nonsmooth nonconvex minimization problem. Further we intend to study the approximation errors for these problems. Of course here the right choice of the norm, proper meshes and stopping criteria will be crutial. In our computations to elastic contact problems we applied efficiently our theory developed in Section 4 in combination with the decomposition $f = f_1 + f_0$ as suggested in in Section2.3.

8 Appendix

8.1 Estimates for the First Eigenvalue of the 1-Laplace operator

In the following we give upper estimates for the first eigenvalue of the 1-Laplace operator for $n = 3$. We use the notation of Section 6.5. The aim is to compute $E_1\left(\chi_{\Omega^*_{r_m}}\right)$ for some $\Omega \subseteq \mathbb{R}^3$.

8.1.1 The Cube

For $\Omega =]0, a[^3$ we observe for the perimeter and the area of Ω_r^* that:
The area $va(r)$ is the area at the 6 faces plus the area of the 12 edges (which give 3 times the lateral area of a cylinder) plus the area in the corners (which is the area of a sphere.) Similar we have that the

volume $vo(r)$is the sum of the volume of the cuboids at the 6 faces, the volume of 3 cylinders at the 12 edges plus the volume of the ball in the corners plus the volume of the inner square.
So we conclude:

$$\begin{aligned} P(\Omega_r^*) &:= 6(a-2r)^2 + 6\pi r(a-2r) + 4\pi r^2 \,, \\ A(\Omega_r^*) &:= 6r(a-2r)^2 + 3\pi r^2(a-2r) + \frac{4}{3}\pi r^3 + (a-2r)^3 \,. \end{aligned}$$

$P(\Omega_r^*)$ and $A(\Omega_r^*)$ we can compute easily numerically. On can plot the grapf of $\phi : r \mapsto \frac{P(\Omega_r^*)}{A(\Omega_r^*)}$ we observe that ϕ is decreasing on $]0, r_m[$ and increasing on $I_r \backslash]0, r_m[$, where $r_m \approx 0.525$ and

$$\frac{P(\Omega_{r_m}^*)}{A(\Omega_{r_m}^*)} \approx 2.7 \,.$$

8.1.2 The Cylinder

Also for the cylinder Ω given

$$\Omega_{a,b} := \left\{ \Big(x, y, z\Big) \in \mathbb{R}^3 \;\middle|\; x^2 + y^2 < a^2 \;\text{ and }\; z \in]0, b[\right\} \,. \tag{8.1}$$

one can determine with little effort Ω_r^*, because Ω_r^* is the union of cylinders and the outer parts of tori. For the concrete formulae of the volumes and areas of those sets we refer to [42] again. For fixed $a, b > 0$ we obtain

$$\begin{aligned} P(\Omega_r^*) &= 2\pi(a-r)^2 + 2\pi a(b-2r) + 2\pi r\left(\pi(a-r) + 2r\right) \\ A(\Omega_r^*) &= 2\pi(a-r)^2 r + \pi a^2(b-2r) + 2\pi r^2 \left((a-r)\frac{\pi}{2} + \frac{2}{3}r\right) \,. \end{aligned}$$

Now one can compute r_m and $E_1\Big(\chi_{\Omega_{r_m}^*}\Big)$ numerically. We refer to Section 6.5 for the results.

8.2 Approximation by Continuous Functions

We will execute here the missing computations of Section 6.6. Let $U_1 := B_{\mathbb{R}^2}(0\,,\,1)$ and let U_2 be the Cheeger set of $]-0.5,\, 0.5[^2$. Further let v_U be defined by (6.68) for oben $U \subset \mathbb{R}^2$.

Since the volume of a cone with radius r and high $\tilde{h}$ is given by $\frac{\pi}{3} r^2 \tilde{h}$, cf. [42], we have

$$G_1(v_{U_1}^\varepsilon) = \frac{\pi}{3} 1^2 \frac{1}{\varepsilon} - \frac{\pi}{3}(1-\varepsilon)^2 \left(\frac{1}{\varepsilon} - 1\right) = \pi\left(1 - \varepsilon + \frac{\varepsilon^2}{3}\right) \,.$$

Further we observe $|Dv_{U_1}^\varepsilon|(x) = \frac{1}{\varepsilon}$ on $B_{\mathbb{R}^2}(0\,,\,1) \setminus B_{\mathbb{R}^2}(0\,,\,1-\varepsilon)$ and 0 else. Thus

$$F_1(v_{U_1}^\varepsilon) = |B_{\mathbb{R}^2}(0\,,\,1) \setminus B_{\mathbb{R}^2}(0\,,\,1-\varepsilon)\,| \,\frac{1}{\varepsilon} = \pi(1^2 - (1-\varepsilon)^2)\frac{1}{\varepsilon} = \pi(2-\varepsilon) \,.$$

Therefore we can analytically determine

$$E_1(v^\varepsilon_{U_1}) := \frac{F_1(v^\varepsilon_{U_1})}{G_1(v^\varepsilon_{U_1})} = \frac{2-\varepsilon}{1-\varepsilon+\frac{\varepsilon^2}{3}} \tag{8.2}$$

and observe $\varepsilon \mapsto E_1(v^\varepsilon_{U_1})$ is continuously differentiable for $0 < \varepsilon < 1$ and we can extent to a continuously differentiable function on $[0,1]$ with

$$\lim_{\varepsilon\to 0} E_1(v^\varepsilon_{U_1}) = E_1(\chi_{U_1}) \; .$$

Therefore

$$|E_1(v^\varepsilon_{U_1}) - E_1(\chi_{U_1})| = o(\varepsilon) \; .$$

On observes further

$$\left\|v^\varepsilon_{U_1} - \chi_{U_1}\right\|_{L_1} \leq |B_{\mathbb{R}^2}\,(0\,,\,1) \setminus B_{\mathbb{R}^2}\,(0\,,\,1-\varepsilon)\,| = \pi(1^2-(1-\varepsilon)^2) = \varepsilon\pi(2-\varepsilon) = o(\varepsilon) \; .$$

Next we consider U_2. Observe

$$U_2 = \bigcup_{x\in]0,1[^2:\; dist_{\partial C}(x)>r} B_{\mathbb{R}^2}\,(x\,,\,r) \quad \text{with } r := \frac{1}{\lambda_1},$$

where λ_1 is the first eigenvalue of the 1 Laplace operator, cf. [32]. Thus we obtain for U_2:

$$\begin{aligned}
G_1(v^\varepsilon_{U_2}) &= (a-2r)^2 + 4(a-2r)(r-\varepsilon) + 4(a-2r)\frac{\varepsilon}{2} + \frac{\pi}{3}r^2\frac{r}{\varepsilon} - \frac{\pi}{3}(r-\varepsilon)^2\left(\frac{r}{\varepsilon}-1\right) \\
&= (a-2r)^2 + 4(a-2r)(r-\frac{\varepsilon}{2}) + \pi\left(r^2 - r\varepsilon + \frac{\varepsilon^2}{3}\right) \; , \\
F_1(v^\varepsilon_{U_2}) &= 4(a-2r) + \pi(r^2-(r-\varepsilon)^2)\frac{1}{\varepsilon} = 4(a-2r) + \pi(2r-\varepsilon) \; , \\
E_1(v^\varepsilon_{U_2}) &= \frac{4(a-2r)+\pi(2r-\varepsilon)}{(a-2r)^2 + 4(a-2r)(r-\frac{\varepsilon}{2}) + \pi\left(r^2 - r\varepsilon + \frac{\varepsilon^2}{3}\right)} \; .
\end{aligned}$$

So we obtain again

$$|E_1(v^\varepsilon_{U_2}) - E_1(\chi_{U_2})| = o(\varepsilon) \quad \text{and} \quad \left\|v^\varepsilon_{U_2} - \chi_{U_2}\right\|_{L_1} = o(\varepsilon) \; .$$

8.3 Further Results

Lemma 8.3 *$F_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ and $G_p : W_0^{1,p}(\Omega) \to \mathbb{R}$ are Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$.*

Proof. The composition of Fréchet differentiable functions is Fréchet differentiable, cf. [12, Theorem 1.1.14]. Further, continuous linear mappings are Fréchet differentiable. Therefore, if we prove Fréchet differentiable of a function f with respect to some norm, then f is Fréchet differentiable with respect to every equivalent norm and with respect to every stronger norm.

- W.l.o.g. we consider on $W_0^{1,p}(\Omega)$ the norm $\|\cdot\| := \|\nabla\cdot\|_{L^p(\Omega)}$. Thus, the Clarkson inequalities

hold on $W_0^{1,p}(\Omega)$, cf. Lemma 3.26. Thus, the Clarkson inequalities hold on $W_0^{1,p}(\Omega)'$ too , cf. Lemma 3.27. The Clarkson inequalities give directly that for every sequences $(u_k')_{k\in\mathbb{N}}$ and $(v_k')_{k\in\mathbb{N}}$ in $W_0^{1,p}(\Omega)'$ with

$$\|u_k'\| = \|v_k'\| = 1 \quad \text{for every } \ k \in \mathbb{N} \ \text{ and } \ \|u_k' + v_k'\| \to 2 \ \text{ as } k \to \infty$$

holds

$$\|u_k' - v_k'\| \to 0 \ \text{ as } k \to \infty \ ,$$

i.e. $W_0^{1,p}(\Omega)'$ is uniformly convex, cf. [12, Definition 2.2.1]. Now, [12, Theorem 2.2.14] gives that $W_0^{1,p}(\Omega)$ is uniformly smooth. Therefore [12, Theorem 1.3.12] tells us that the norm $\|\cdot\|$ is Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$. Thus $F_p = \|\cdot\|^p$ is Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$.

- Analogously we prove that the L^p norm is is Fréchet differentiable on $L^p(\Omega) \setminus \{0\}$. The identity operator $Id : W_0^{1,p}(\Omega) \to L^p(\Omega)$ is linear and continuous. Hence, the L^p norm is Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$ with respect to the Sobolev norm. So we obtain again that $G_p = \|\cdot\|_{L^p(\Omega)}^p$ is Fréchet differentiable on $W_0^{1,p}(\Omega) \setminus \{0\}$.

$\diamondsuit$

Nomenclature

Mappings

A Area, page 130

χ_C Characteristic function of C

χ_C^a Piecewise affine approximation of χ_C, page 155

$dist$ Distance function, page 50

E_p Energy function, page 129

$\tilde{E}_p$ Energy function, page 129

$f : X \to \mathbb{R}$ Lipschitz continuous mapping (if nothing else said)

F_1 Total variation, page 129

F_p Energy function, page 128

G_1 L^1 norm, page 129

$f^0(x; v)$ Clarke's generalized directional derivative, page 11

$\partial f(x)$ Clarke's generalized gradient of f at x, page 12

f', f_0' Normally those denote elements of $\partial f(x)$ or $\partial^\varepsilon f(x) \subseteq X'$

$D_A f(x)$ Representation of a generalized gradient with respect to $\|\cdot\|_A$, page 85

$f'(x; v)$ One-sided directional derivative, page 14

G_p Constraint function, page 128

G_p^a Approximation of G_p, page 135

$\mathcal{L}^n$ Hausdorff measure

j Dual mapping, page 26

$\hat{DF}$ Approxiamtion of classical Jacobian of F

DF Classical Jacobian of F

M Representation of an element of the generalized Jacobian , page 86

$\hat{M}$ Approximation of M , page 89

$\|\cdot\|_X$ Norm in the Banach space X

$\|\cdot\|_A$ Equivalent norm on the Hilbert space H , page 85

$\|\cdot\|_\infty$ Supremum norm

P Perimeter, page 130

$R : H' \to H$ Riesz mapping $R : H' \to H$ for a Hilbert space H, which allows to identify H and H', cf. [59].

$(x_k)_{k\in\mathbb{N}}$ Sequence with elements x_k

$\langle \cdot \,|\, \cdot \rangle_H$ Scalar product in Hilbert space H

$\langle \cdot \,|\, \cdot \rangle_A$ Equivalent scalar product in Hilbert space $\mathbb{R}^n$, page 85

$\sqrt{A}$ Square root of a symmetric, positively definite matrix A , page 85

TV Total variation, page 126

$f_\varepsilon^0(x;d)$ Generalized directional derivative of f on $\overline{B_X\,(0\,,\,\varepsilon)}$, page 29

Constants

K_{eq} Constant which gives the equivalence of the norms, page 55

λ_p First eigenvalue of the p-Laplace operator, page 128

Sets

A^B Set of all mappings from B to A

∂A Boundary of A

$\overline{A}$ Closure of A

$conv(A)$ Convex hull of A

$B_X\,(x_0\,,\,r)$ Open ball in X with center x_0 and radius r

C Cheeger set

$\partial^{conv} f\,(A)$ Generalized gradient of f on A, page 29

$\partial^\varepsilon f(x)$ Generalized gradient of f on $\overline{B_X\,(0\,,\,\varepsilon)}$, page 29

$\partial F(x)$ Generalized Jacobian of F at x , page 86

$[x, y]$ Closed line segment $conv\{x, y\}$

(x, y) Open line segment $(conv\{x, y\}) \setminus \{x, y\}$

$M_{<\vartheta(x)}$ Set to avoid dividing by zero, page 52

$M^k_{<\vartheta}$ Set to avoid dividing by zero, depending on k, page 55

$m : X \to \mathbb{R}$ Model function of f, page 51

Ω Domain in $\mathbb{R}^n$

$\prod_{i=1}^{m} A_i$ Cartesian product of the sets A_i

$S_X(x_0, r)$ Boundary of open ball in X with center x_0 and radius r

τ_h Mesh on Ω consisting of regular triangles, page 135

$U(x)$ Neighborhood of x

Acronyms

LSRS Linesearch with restart procedures, page 110

PGM Projected Gradient Method, page 156

SDM Steepest Descent Method, page 156

Points

d_k Descent direction in iteration k

$x_{\min}$ Minimizer of the function f

u_k Iteration point/function in iteration k

u_0 Initial function

x_k Iteration point in iteration k

x_0 Initial point

Spaces

$BV(\Omega)$ Functions in $L^1(\Omega)$ with bounded variation

$C_0^\infty(\Omega, \mathbb{R})$ Space of infinitely often differentiable functions from Ω to $\mathbb{R}$ with compact support

$GL(\mathbb{R}^n)$ General linear group of regular matrices in $\mathbb{R}^{n\times n}$

H Hilbert space

$L^p(\Omega)$ Space of p-integrable functions on Ω

$W^{k,p}(\Omega)$ Sobolev space of all function in $L_p(\Omega)$ which admit the k-th weak derivative

$W_0^{k,p}(\Omega)$ Closure of $C_0^\infty(\Omega, \mathbb{R})$ in $W^{k,p}(\Omega)$

V_h Space of continuous, piecewise affine functions related to the mesh τ_h, page 135

X Banach space

X' Dual space of Banach space X equipped with the induced norm

X'' Bidual space of Banach space X equipped with the induced norm

References

[1] W. Alt: Nichtlineare Optimierung, Eine Einführung in Theorie, Verfahren und Anwendungen, Vieweg Verlag, Wiesbaden, (2002)

[2] W. Alt: Numerische Verfahren der konvexen, nichtglatten Optimierung, Eine anwendungsorientierte Einführung, Teubner Verlag, Wiesbaden, (2004)

[3] J. Aubin, I. Ekeland: Applied Nonlinear Analysis, Wiley, New York (1984)

[4] J. V. Burke, A. S. Lewis and M. L. Overton: A Robust Gradient Sampling Algorithm for Nonsmooth, Nonconvex Optimization, SIAM J. OPTIM., Vol. 15,No. 3, (2005) pp.751-779

[5] H. Bauer: Wahrscheinlichkeitstheorie, 5 Auflage, Walter de Gruyter, Berlin- New York, (2002)

[6] A. CAUCHY: Méthode générale pour la résolution des systémes d'éqations simultanies. Comptes Rendus 25, 2, 536 (1847)

[7] E. Cheney: Introduction to Approximation Theory, McGraw-Hill Book Company, (1966)

[8] R. Chill and E. Fasangova: Gradient Systems, Lecture Notes of the 13th International Internet Seminar, Matfyzpress, Prague, (2010)

[9] K.C. Chang: Variational methods for non-differentiable functionals and their applications to partial differential equations, J. Math. Anal. Appl. 80, 102129 (1981)

[10] P.G. Ciarlet, Mathematical Elasticity, Three Dimensional Elasticity, vol. 1, North Holland, Amsterdam, (1988)

[11] P.G. Ciarlet, J.L. Lions: Handbook of Numerical Analysis, Finite Element Methods (Part 1), Volume II, North Holland, Amsterdam, The Netherlands, (1991)

[12] I. Cioranescu: Geometry of Banach Spaces, Duality Mappings and Nonlinear Problems, Kluwer Academic Publishers, Dordrecht, The Netherlands, (1990)

[13] F. H. Clarke : Optimization and Nonsmooth Analysis, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, (1990)

[14] J. A. Clarkson: Uniformly convex spaces. Transactions of the American Mathematical Society, Band 40 (1936), pp. 396-414.

[15] A. R. Conn, N. I. M. Gould, P. L. Toint: Trust-Region Methods. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, (2000)

[16] F. Demengel, F. De Vuyst, M. Morton: A Numerical Approach for computing the First Eigenvalue of the 1-Laplacian of the Square and Other Particular Sets, Preprint (2002)

[17] J. E. Dennis Jr., S. B. Li, R. A. Tapia: A unified approach to global convergence of trust region methods for nonsmooth optimization, Mathematical Programming 68 (1995), pp. 319-346

[18] I. Ekeland, R. Temam: Convex Analysis and Variational Problems, North-Holland Publ. Co., Amsterdam, (1976)

[19] L. Evans, R. Gariepy: Measure Theory and Fine Properties of Functions, CRC Press, Boca Raton, (1992)

[20] A. Fischer: Solution of monotone complementarity problems with locally lipschitzian functions, Math. Programming, 76 (1997), pp. 513532.

[21] H. Goering, H.-G. Roos, L. Tobiska: Die Finite-Elemente-Methode für Anfänger. Wiley, Berlin, (2010)

[22] A.A. Goldstein: Cauchy's method of minimization, Numerische Mathematik 4 (1962), pp. 146-150,

[23] A.A. Goldstein: On steepest descent, SIAM Journal on Control Ser. A, 3 (1) (1965), pp. 147-151.

[24] A.A. Goldstein: Optimization of Lipschitz continuous functions, Math. Program., 13 (1977), pp. 1422

[25] C. Grosan, A. Abraham: A Novel Global Optimization Technique for High Dimensional Functions, International Journal of Intelligent Systems, 24(4) (2009), pp. 421 - 440

[26] L. Hörmander: Sur la fonction d'appui des ensembles convexes dans un espace localement convexe, Ark. Math. **3**, 181-186

[27] C. Horgan, G. Saccomandi: Constitutive Models for Compressible Nonlinearly Elastic Materials with Limiting Chain Extensibility, Journal of Elasticity, Volume 77, Number 2, (November 2004), pp. 123-138.

[28] J. Horák : Constrained mountain pass algorithm for the numerical solution of semilinear elliptic problems, Numer. Math. 98 (2004), no.2, pp. 251-276

[29] J. Horák : Numerical Investigation of the Smallest Eigenvalues of the $p-$Laplace Operator on Planar Domains, Electronic Journal of Differential Equations. (2011), Vol. 2011, Special section, pp. 1-30.

[30] C. Kanzow: Numerik linearer Gleichungssysteme, Direkte und iterative Verfahren, Springer-Verlag, Berlin Heidelberg, (2005)

[31] M. Kato,Y. Takahashi: Type, Cotype Constants and Clarkson's Inequalities for Banach spaces, Math. Nachr. 186 (1997), pp 187-195.

[32] B.Kawohl, T. Lachand-Robert: Characterization of Cheeger sets for convex subsets of the plane, Pacific J. Math., 225 (1) (2006), pp. 103–118.

[33] B. Kawohl, V. Fridman: Isoperimetric estimates for the first eigenvalue of the p-Laplace operator and the Cheeger constant, Comment. Math. Univ. Carolinae, 44 (2003), pp. 659-667.

[34] B. Kawohl, M. Novaga: The p-Laplace eigenvalue problem as p approaches 1 and Cheeger sets in a Finsler metric, J. Convex Anal. 15 (2008), pp. 623-634.

[35] B. Kawohl, F. Schuricht: Dirichlet problems for the 1-Laplace operator, including the eigenvalue problem. Comm. Contemp. Math. 9 (2007), pp. 1-29

[36] K. C. Kiwiel: Convergence of the Gradient Sampling Algorithm for Nonsmooth Nonconvex Optimization, SIAM J. OPTIM., Vol. 18,No. 2, (2007), pp.379-388

[37] R. Krause: A non-Smooth Multiscale Method for Solving Frictional Two-Body Conctact Problems ind $2d$ and $3d$ with Multigrid Efficiency, SIAM Journal on Scientific Computing Vol. 31, No. 2: (2009), pp.1399-1423

[38] Y. W. Leung, Y. Wang: An orthogonal genetic algorithm with quantization for global numerical numerical optimization, IEEE Trans. Evol. Comput.,vol. 5, Feb. (2001), pp. 4153

[39] A. S. Lewis, M. L. Overton: Nonsmooth optimization via BFGS. http://www.cs.nyu.edu/overton/papers/pdffiles/bfgs_inexactLS.pdf (2008)

[40] S.Littig, F. Schuricht: Convergence of the eigenvalues of the p-Laplace operators as p goes to 1, Calc. Var. Partial Differential Equations, 49 (2014), pp. 707-727

[41] J. M. Martinez, A. C. Moretti: A trust region method for minimization of nonsmooth functions with linear constraints, Mathematical Programming 76 (1997), pp. 431-449

[42] G. Meriger, G. Mühlbach, D. Wille, T. Wirth: Formeln + Hilfen zur Höheren Mathematik, 7 Auflage, Binomi Verlag, 2013

[43] R. Mifflin: Semismooth and semiconvex functions in constrained optimization, SIAM J. Control Optim., 15 (1977), pp. 959972.

[44] J. Nocedal, S. J. Wright : Numerical Optimization, Springer-Verlag, New York, (2006)

[45] R.W. Ogden: Nonlinear Elastic Deformations, Dover Publications Inc. Mineola, New York, (1998)

[46] J.-S. Pang, L. Qi : Nonsmooth equations: motivation and algorithms, SIAM J. Optim., 3 (1993), pp. 443465.

[47] L. Qi, J. Sun: A nonsmooth version of Newtons method, Math. Programming, 58 (1993), pp. 353367.

[48] L. Qi, J. Sun: A trust region algorithm for minimization of locally Lipschitzian functions, Mathematical Programming, 66 (1994), pp. 25-43

[49] S.M. Robinson: Newtons method for a class of nonsmooth functions, Set-Valued Anal., 2 (1994), pp. 291305.

[50] R. T. Rockafellar: Convex Analysis, Princeton University Press, Princeton, NJ, (1970)

[51] W. Rudin: Reelle und Komplexe Analysis, McGraw-Hill, Inc., New York, (1987)

[52] H. Schramm: Eine Kombination von Bundle- und Trust-Region-Verfahren zur Lösung nichtdifferenzierbarer Optimierungsprobleme, Bd. 30 d. Reihe Bayreuther Mathematischer Schriften, Universität Bayreuth, (1989)

[53] F. Schuricht: Minimax principle for eigenvalue variational inequalities in the nonsmooth case, Math. Nachr., 152(1991), pp. 21143

[54] E. Spedicato: Computational Experience with Quasi-Newton Algorithms for Minimization Problems of Moderately Large Size, Report CISE-N-175International Press, Milano (1975) Segrate

[55] P. Spellucci: Numerische Verfahren der nichtlinearen Optimierung, Birkhäuser Verlag, Basel, (1993)

[56] M. Ulbrich: Nonsmooth Newton-like Methods for Variational Inequalities and Constrained Optimization Problems in Function Spaces, Habilitation, Munich (2002)

[57] M. Ulbrich: Semismooth Newton Methods for Operator Equations in Function Spaces. SIAM Journal on Optimization 13:3 (2006), pp. 805-841.

[58] J. Werner: Numerische Mathematik, 2. Vieweg, Braunschweig, (1992)

[59] D. Werner: Funktionalanaylsis, 5 erweiterte Auflage, Springer Verlag, Berlin Heidelberg, (2005)

[60] E. Zeidler: Nonlinear Functional Analysis and its Applications III, Variational Methods and Optimization, Springer-Verlag, New York, New York, (1985)

[61] E. Zeidler: Nonlinear Functional Analysis and its Applications I, Fixed -Point Theorems, Springer-Verlag, New York, New York, (1986)

Versicherung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe; die aus fremden Quellen direkt oder indirekt bernommenen Gedanken sind als solche kenntlich gemacht. Die Arbeit wurde bisher weder im Inland noch im Ausland in gleicher oder ähnlicher Form einer anderen Prüfungsbehörde vorgelegt.

Datum, Unterschrift